REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
24-07-2015	MS Thesis	-

4. TITLE AND SUBTITLE
On the performance of the Underwater Acoustic Sensor Networks

5a. CONTRACT NUMBER
W911NF-11-1-0144

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
206022

6. AUTHORS
Mahmoud Elsayed

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
The University of the District of Columbia
Computer Science and Informati
Briana Lowe Wellman
Washington, DC 20008 -1122

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
58962-NS-REP.19

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
In this thesis, we propose an experimental approach to enhance the performance of the underwater acoustic sensor networks and to decrease the network power consumption. Distributed Underwater Wireless Sensors Network (UWSN), based on acoustic waves, is an area of active research in underwater communications. UWSN is widely used in applications, such as military surveillance, disaster monitoring and ocean observation. UWSN has many constraints mainly due to limited capacity, propagation loss, as well as power limitation since in underwater environment solar energy cannot be used to recharge batteries. In our approach, we estimate the number of

15. SUBJECT TERMS
underwater communication, wireless sensors, mutual information

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Paul Cotae
UU	UU	UU	UU		19b. TELEPHONE NUMBER 202-274-6290

Report Title

On the performance of the Underwater Acoustic Sensor Networks

ABSTRACT

In this thesis, we propose an experimental approach to enhance the performance of the underwater acoustic sensor networks and to decrease the network power consumption. Distributed Underwater Wireless Sensors Network (UWSN), based on acoustic waves, is an area of active research in underwater communications. UWSN is widely used in applications, such as military surveillance, disaster monitoring and ocean observation. UWSN has many constraints mainly due to limited capacity, propagation loss, as well as power limitation since in underwater environment solar energy cannot be used to recharge batteries. In our approach, we estimate the number of operating receivers within the network based on the knowledge of the Bit-Error-Rate (BER) and the signal detection of the transmitted message, and, therefore, take advantage of the BER variation, which depends on the underwater acoustic channel environment. In our experimental observations, we considered two network models: multi hop relay acoustic sensors network and parallel acoustic sensors network.

In the second part of the research, we use the information theoretic aspects to analyze and study the impact of using mobile/moving sensor nodes, as opposed to fixed nodes in UWSN. The jump from fixed sensor nodes to moving sensor nodes enables many advantages in UWSN, but also introduces several complications. The two main significant advantages of using mobile sensor nodes over fixed nodes are; a) the reduction of sensor nodes needed, and b) the ability to handle dynamic situation. Thus, the same work can be accomplished by fewer mobile nodes instead of a large number of fixed sensor nodes. Also, we developed software “SAM Control” written in MATLAB programming language to facilitate our experiments. SAM Control coordinates and manages the communications between the acoustic modems used in our experiments, while at the same time allowing data gathering of all experimental observations and performance analysis of power consumption, BER, mutual information, and information loss. Furthermore, simulations study of the underwater acoustic channel model is also presented.

On the performance of the Underwater Acoustic Sensor Networks

By
Mahmoud Elsayed

A thesis submitted to Graduate Faculty of the
University of the District of Columbia
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in
Electrical Engineering

Washington, DC
May, 2015

On the Performance of Underwater Acoustic Sensor Networks

In this thesis, we propose an experimental approach to enhance the performance of the underwater acoustic sensor networks and to decrease the network power consumption. Distributed Underwater Wireless Sensors Network (UWSN), based on acoustic waves, is an area of active research in underwater communications. UWSN is widely used in applications, such as military surveillance, disaster monitoring and ocean observation. UWSN has many constraints mainly due to limited capacity, propagation loss, as well as power limitation since in underwater environment solar energy cannot be used to recharge batteries. In our approach, we estimate the number of operating receivers within the network based on the knowledge of the Bit-Error-Rate (BER) and the signal detection of the transmitted message, and, therefore, take advantage of the BER variation, which depends on the underwater acoustic channel environment. In our experimental observations, we considered two network models: multi hop relay acoustic sensors network and parallel acoustic sensors network.

In the second part of the research, we use the information theoretic aspects to analyze and study the impact of using mobile/moving sensor nodes, as opposed to fixed nodes in UWSN. The jump from fixed sensor nodes to moving sensor nodes enables many advantages in UWSN, but also introduces several complications. The two main significant advantages of using mobile sensor nodes over fixed nodes are; a) the reduction of sensor nodes needed, and b) the ability to handle dynamic situation. Thus, the same work can be accomplished by fewer mobile nodes instead of a large number of fixed sensor nodes. Also, we developed software “SAM Control” written in MATLAB programming language to facilitate our experiments. SAM Control coordinates and manages the communications between the acoustic modems used in our experiments, while at the same time allowing data gathering of all experimental observations and performance analysis of power consumption, BER, mutual information, and information loss. Furthermore, simulations study of the underwater acoustic channel model is also presented.

Acknowledgments

I would like to express my deepest gratitude to my mentor and advisor Dr. Paul Cotae, P.I. of the Department of Defense (DoD) Research Grant W911NE-11-1-0144, for his guidance and financial support throughout my master study and research.

My sincere thanks go to the Naval Research Laboratory, in particular to Dr. Ira S. Moskowitz for his useful remarks and for his research on top of which my thesis is built.

I would like to acknowledge the Electrical and Computer Engineering Department which has been helpful throughout my master degree, especially Dr. Sasan Haghani, Dr. Samuel Lakeou, Dr. Zhang, and Dr. Mahmoud.

I would also like to thank my colleagues, Juan Jfrroniac, Dan Albano, and Moustafa Sayed for sharing their knowledge and assisting me in developing the software as well as conducting my experiments.

Last but not least, my sincere thanks to my Mom, Cousin, my sister-in-law (Reem), my brother and sister for their unending love, support and guidance.

List of Publications

1. Raju Shrestha, Mahmoud Elsayed, P. Cotae, "On the Mutual Information of Sensor Networks in Underwater Wireless Communication: An Experimental Approach" *ASEE Zone I Conference, April 3-5, 201*4, University of Bridgeport, Bridgeport, CT, USA.

2. Mahmoud Elsayed, P. Cotae, "On the Performance of the Underwater Wireless Communication Sensor Networks: Work in Progress" *ASEE Mid-Atlantic Fall 2014 Conference, November 14- 15, 2014, Swarthmore College, Swarthmore, PA, USA.*

3. Mahmoud Elsayed, P. Cotae, I. Maskowitch "On the Performance of the Underwater Wireless Sensor Networks: An Experimental Approach" *Proceedings of the ASEE Gulf-Southwest Annual Conference, March 25-27, the University of Texas at San Antonio, San Antonio, Taxes, USA.*

In Submission

1. Mahmoud Elsayed, P. Cotae, I. Maskowitch "On the Performance of the Underwater Wireless Sensor Networks: An Experimental Approach*" ASEE 2015 North-East Section Regional Conference, April 30 - May 2, 2015, Northeastern University, Boston, Massachusetts, USA.*

Contents

On the Performance of Underwater Acoustic Sensor Networks i
Acknowledgments ii
List of Publications iii
Contents ..... **Error! Bookmark not defined.**
List of Figures vi
List of Tables vii
List of Notations viii
List of Abbreviations x
Summary xii
Chapter 1 14
Introduction 14
1.2 Thesis Significance, contributions and findings 15
1.3 Thesis Structure 17
Chapter 2 18
Simulation study of the UAC 18
2.1 Underwater Chanel Model 18
2.2 Signal Attenuation 19
2.3 The Noise Model in UWSNs 21
2.4 Channel Capacity 22
2.5 Conclusion 23
Chapter 3 24
Modulation Techniques 24
3.1 PN-sequence modulation 24
3.2 Direct Sequence Spread Spectrum (DSSS)/ Binary Phase Shift Keying (BPSK) 25
Chapter 4 27
Information Theoretic Approaches 27
Entropy, joint and Conditional entropy 27
4.1.1 Entropy 27
4.1.2 Joint Entropy 27
4.1.3 Conditional Entropy 28
4.2 Mutual information 28

Information Loss 29
4.3 Bit Error Rate (BER) 29
BER and Eb/No 29
Factors affecting BER 30
4.4 Optimum Channel Capacity 30
4.5 Mutual Information over Cross correlation 31
4.6 Relating Information Theory to our Network Model 31
4.6.1 Collaborative Sensor Network system 31
4.6.2 Parallel Sensor Network system 34
4.7 Conclusion 36
Chapter 5 37
Experimental Approach 37
5.1 Hardware and Software Details 37
5.1.1 Equipment 37
5.1.2 Software development (SAM Control) 39
5.2 Collaborative network using fixed sensor nodes 40
Collaborative network results. 41
5.3 Parallel network using fixed sensor nodes. 43
Parallel network results. 44
5.4 UWSN network using moving sensor nodes 44
UWSN network using moving sensor nodes results 45
5.5 Experiment Approach based on the BER. 47
5.5.1 Method 47
5.5.2 Scenario I 48
5.5.3 Scenario 2: Target moving (External noise) 49
Chapter 6 50
Discussion, Results and Conclusion 50
6.2 Conclusion 51
References 54
Appendix 57
MATLAB code for the Information theory aspects tools and Experimental Approach 58
MATLABCode for the GUI (SAM Control) 61

List of Figures

Figure 1: The different aspects of the main project 16
Figure 2**:** Absorption Coefficient 20
Figure 3: Overall transmission loss (f = 50KHz). 20
Figure 4: Power spectral density of the ambient noise using the approximation equation in. 22
Figure 5: BPSK Transmitting channel model 26
Figure 6: BPSK Receiving channel model 26
Figure 7: Graphical representation of Conditional Entropy and MI of X and Y 28
Figure: 8. Cascaded channel model. 32
Figure 9: Parallel sensor network system 34
Figure 10: Pair of the Acoustic Modem used in experiments 38
Figure 11: SAM Control GUI 39
Figure 12: Cascaded underwater wireless sensor network 40
Figure. 13. Parallel underwater wireless sensor network 43
Figure 14: Underwater Mobile Acoustic Sensor Network 45
Figure 15: Experimental approach setup 47
Figure:16. PSD of Noise (No(f)) experimentally measured in underwater 50

List of Tables

Table 1. Underwater Acoustic Communication System Ranges [3] 18
TABLE 2: Specification of the modem 38
Table 3: Bit loss test for different distance of sensors 41
Table 4: Result analysis for Tx1 and Rx1 (d=5 m) 41
Table 5: Result analysis for Tx2 and Rx2 (d=5 m) 42
Table 6: Result analysis for Tx3 and Rx3 (d=5 m) 42
Table 7: Result analysis for Tx1 and Rx4 (d=15 m) 42
Table 8: Result analysis for cascaded sensors network 43
Table 9: Result analysis for Tx1 and Rx1 44
Table 10: Bit loss test for mobile sensor nodes of maximum distances = 15m. 46
Table 11: Bit loss test for mobile sensor nodes moving in vertical direction. 46
Table 12: Performance results of using mobile sensor nodes 46
Table 13: Numerical results for scenario #1 48
Table 14: Numerical results for scenario #2 49
Table 15: Result analysis of using fixed Sensor nodes 51
Table 16: Result analysis of using fixed sensor nodes 51
Table 17: Summary results for using mobile sensor nodes 52
Table 18 (no target is moving): Numerical results for scenario #1 52
Table 19: Approach results in Noisy UAC (Scenario#2) 52

List of Notations

SNR	Signal to Noise Ratio
SNR_o	Signal to Noise Ratio of each node
d	Link distance
C	Channel Capacity
BW	Bandwidth
P	Power of Signal
N_o	Power of Noise
E_S	Signal Energy
E_N	Noise Energy
f	Carrier Frequency
c_s	Sound Speed
dB	Decibel
α	Absorption Coefficient
$\theta_b(k)$	Periodic Autocorrelation
A(r,f)	Path-loss in Underwater Communications Channels
a(f)	Absorption Coefficient
KHz	Kilo Hertz
MC	Monitoring center
$N_o(f)$	Noise in Underwater Communication Cannels
PSD	Power Spectral Density
P_b	Probability of Error
$S_r(f)$	Power Spectral Density of the Received Signal
$P_{e,N}$	BER of *N*-hop link
$P_{e,1}$	BER of each hop
R	Bit rate
b	scaling factor for bandwidth
c	scaling factor for capacity

p	scaling factor for power
N	Number of hops over a link distance
d_p	length of the p-th propagation path
s	shipping activity factor
$I(X;Y)$	Mutual information between X and Y
$H(X)$	Entropy of X
$H(Y\|X)$	Conditional entropy of Y given X
$p(x)$	Probability distribution of X
$p(x, y)$	Joint probability distribution of X and Y

List of Abbreviations

UWC	Underwater Wireless Communication
UAC	Underwater Acoustic Communication
BER	Bit Error Rate
UWSN	Underwater Wireless Sensor Network
SNR	Signal to Noise Ratio
SINR	Signal to Interference and Noise Ratio
BW	Bandwidth
BS	Base station
CH	Cluster Head
BPSK	Binary Phase Shift Keying
PPM	Pulse Position Modulation
PN	Pseudorandom Noise
CDMA	Code Division Multiple Access
MRC	Maximum Ratio Combining
log	Logarithm in base 10
$\log_2$	Logarithm in Base Two
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
PSD	Power Spectral Density
TL	Transmission Loss
AWGN	Additive White Gaussian Noise
DPI	Data Processing Inequality
MAI	Multiple Access Interference

Summary

In this thesis, we use the information theoretic aspects to analyze and study the impact of using mobile/moving sensor nodes, as opposed to fixed nodes in UWSN. The jump from fixed sensor nodes to moving sensor nodes enables many advantages in UWSN, but also introduces several complications. The two main significant advantages of using mobile sensor nodes over fixed nodes are; a) the reduction of sensor nodes needed, and b) the ability to handle dynamic situation. Thus, the same work can be accomplished by fewer mobile nodes instead of a large number of fixed sensor nodes.

Another significant contribution, we propose an experimental approach to enhance the performance of the underwater acoustic sensor networks and to decrease the network power consumption. In our approach, we estimate the number of operating receivers within the network based on the knowledge of the Bit-Error-Rate (BER) and the signal detection of the transmitted message, and, therefore, take advantage of the BER variation, which depends on the underwater acoustic channel environment. Several computer simulations and pool experiments have been carried out to verify our experimental approach. In addition, we estimated a simulation study of the underwater acoustic channel model.

Finally, we developed software "*SAM Control*" written in MATLAB programming language to facilitate our experiments. *SAM Control* coordinates and manages the communications between the acoustic modems used in our experiments, while at the same time allowing data gathering of all experimental observations and performance analysis of power consumption, BER, entropy of message signal, mutual information, and information loss. Our work shows that the use of mobile acoustic sensors in underwater communication system is able to provide a reliable wireless link with low probability of error (<0.003) for different sensor velocities. Moreover, we achieved comparable results in terms of the performance based on information theory aspects.

Chapter 1

Introduction

In any underwater acoustic communication (UAC) system, it is of practical importance to consider environmental effects when we consider the performance measure of signal processing function [1]. There exist various waves for Underwater Wireless Communication (UWC); radio waves, optical waves, and acoustic waves are few to name. Radio waves are good for extra low frequencies (30Hz-300Hz), but are very susceptible to noise and require high transmission power. On the other hand, optical waves suffer from scattering. Therefore, acoustic waves are best for UWC. Underwater Wireless Communication imposes many constraints that affect the design of wireless network like path loss, transmission distance, energy absorption by water, long propagation. After the wide development in wireless digital communication for the underwater environment, it is important to improve the performance of existing system, such as data rate, channel capacity, information loss, probability of error, and power consumption.

In this thesis, we investigate the performance of underwater acoustic sensor networks in underwater wireless communication in terms of mutual information, channel capacity, bit error rate, and information loss based on the transmission power, distance, carrier frequency and the bandwidth of the channel. Moreover, we use the information theoretic aspects to analyze and study the impact of using mobile/moving sensor nodes, as opposed to fixed nodes in UWSN. A placement of multiple acoustic sensors in a multi-hop relay form is needed in numerous applications where data transmission has to be accomplished beyond short distances. This network is effective since the bandwidth of an underwater acoustic communication is severely limited and decays with increase in distances. It is advantageous to accomplish such transmission using sensors in a multi-hop relay form keeping constraints such as transmission rate, transmission delay, Signal-to-Interference and Noise Ratio (SINR) under consideration.

Furthermore, UWSNs are typically characterized by severely energy limited nodes. The reason is the nodes have small batteries and therefore cannot afford much energy to complete tasks. As the life time of any individual sensor in the UWSN is limited, the number sensor nodes that stop

working due to power loss increases with a lengthened deployment time, therefore the coverage area of WSN will shrink. In order to compensate for the power limitation, we present an experimental approach that reduces the power consumption by UWSN. In our approach, we estimate the number of operating receivers within the network based on the knowledge of the Bit-Error-Rate (BER) and the signal detection of the transmitted message, and, therefore, take advantage of the BER variation, which depends on the underwater acoustic channel environment.

1.2 Thesis Significance, contributions and findings

This work is supported by the grant titled "Information - Driven Doppler Shift Estimation and Compensation Methods for Underwater Wireless Sensor Networks", which has different research sections [2], Figure 1. In the main task mentioned above, noncoherent communications methods at the physical layer for target tracking and information theoretic function tools were used to predict the next target position being investigated.

In the sensor system architecture presented in Figure 1, the underwater sensor nodes with acoustic modems are distributed in the underwater space. Each underwater sensor node can measure a specific bit rate of the data around it. The sensors are positioned in clusters and they communicate with the Command and Control center via the Monitoring center (MC). The MC will eventually communicate with Command and Control unit on the shipboard through high capacity wavelength optical fiber links.

Different goals in this project can be summarized as [2]:

1) To analyze and improve the non-data aided methods for Doppler shift estimation and compensation (squaring time phase recovery method, power spectrum method, and partial autocorrelation method);
2) To determine the necessary quantity and deployment strategy of a sensors in a given region and to provide a required security level;
3) To minimize the probability of false alarm and the bit error rate;
4) To improve decision-making on target detection with sensor collaboration in the context of blind Doppler shift estimation and detection, and
5) To improve the ability of a distributed sensor system to detect intruders (targets) and determine how the network wireless sensor design is affected.

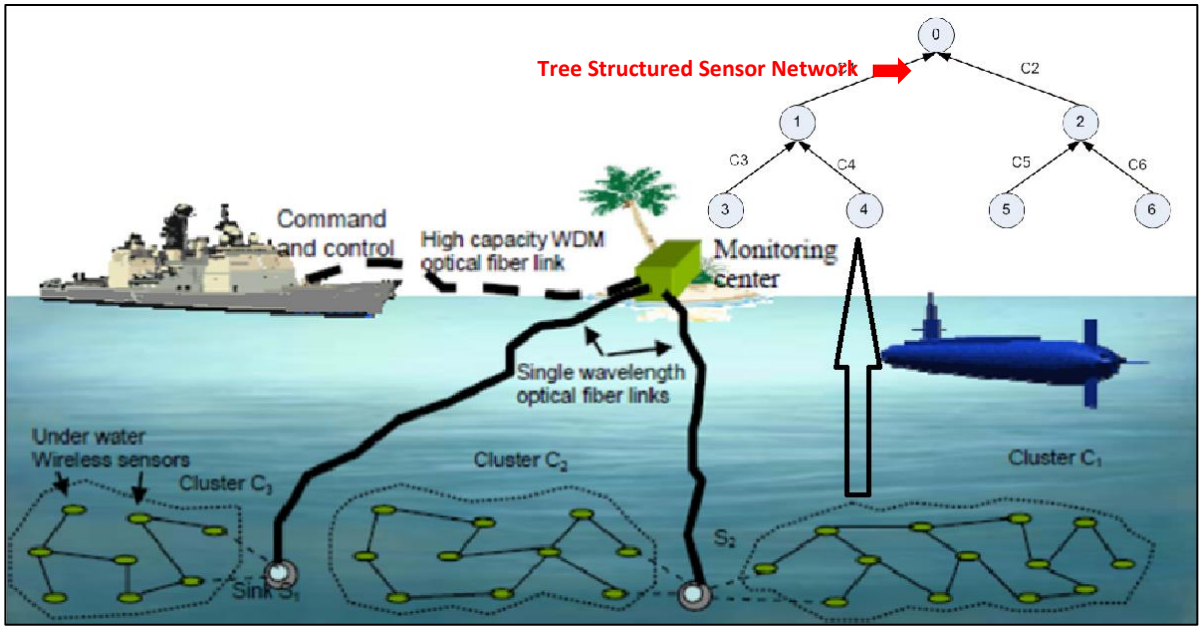

Figure 1: The different aspects of the main project

Our contribution is in the performance of the UWSN and how to practically apply our knowledge and education background to enhance the capabilities of UWSN shown in the figure above. In order to improve the ability of a distributed sensor system, we concentrate on improving mutual information between two data signals in terms of minimum bit error rate, maximum data rate communication, and channel performance. We focus on sensor placement in multi-hop to minimize the bandwidth consumption and mitigate the risk of acoustic link failure and information loss. In order to reduce the communication load for each sensor we develop new distributed algorithms for underwater communication similar to those developed in wireless communication in [3]. The advantage of this sensor placement will help in fulfilling the increasing demand for reliable maximum data rate wireless communication links to accommodate the wide range of underwater application.

1.3 Thesis Structure

To understand the underwater acoustic communication, its channel capacity, information loss, and signal power allocation; one requires knowledge of information theoretic tools, digital communication, and signal processing. Also, we have extensively used the MATLAB programming in system design and simulation of sensors placement to maximize the data rate, mutual information, and channel capacity. It is also very important to explain a few details in appreciable the thesis.

The rest of the thesis is structured is structured as follows:

Chapter 2 introduces the simulation study of the Underwater Acoustic Channel Model (UAC).

Chapter 3 provides the detail about the modulation techniques used in our research project.

Chapter 4 explains the information theoretic approaches and methods we use in evaluating and enhancing our proposed model of multi-hop relay network in underwater communication for using mobile and fixed sensor nodes.

Chapter 5 presents our experimental approach and our methodology based on Bit Error Rate and the diversity technique to enhance the performance of UWSN.

Chapter 6 Summarize our results and conclude the main objective and findings in this thesis.

Chapter 2

Simulation study of the UAC

In this chapter, a simulation study of underwater acoustic propagation loss (TL) and ambient noise will be presented. In addition, we review path-loss, noise in narrowband underwater communication channel, and the channel capacity.

2.1 Underwater Chanel Model

An underwater communication channel is very complex and communicates through propagation of sound, called acoustic communication. Underwater acoustic sensor networks are then formed by acoustically connected bottom sensor nodes, autonomous underwater vehicles, and surface stations that serve as gateways and provide radio communications link to on-shore stations. Underwater acoustic sensor networks consist of sensor nodes and vehicles deployed underwater and networked via the acoustic links to perform collaborative monitoring tasks. However, acoustic channels impose many constraints that affect the design of UWC systems. These are characterized by path loss that depends on both the transmission distance and the signal frequency. The signal frequency determines the absorption loss, which increases the distance as well, eventually imposing a limit on the available bandwidth.

Table 1. Underwater Acoustic Communication System Ranges [4]

Distance	Range(Km)	Bandwidth (KHz)
Short	0.1	>100
Very short	0.1-1	20-60
Medium	1-15	10
Long	15-100	2-6
Very long	1000	<1

Table I shows typical bandwidth of the underwater channel for different ranges. Long-range systems that operate over kilometers may have a bandwidth of only a few kHz, while a short-range system operating over few meters may have more than a hundred kHz bandwidth. This is true because carrier frequency is higher for short range acoustic communication which will give a wide bandwidth to transmit the signal. Likewise, for long range acoustic communication, carrier frequency is lowered and available bandwidth is small.

2.2 Signal Attenuation

Acoustic path loss depends on the signal frequency and the distance between transmitter and receiver. This dependence is a consequence of absorption loss (i.e. transfer of acoustic energy into heat) [4].

$$A(d,f) = d^{\alpha} a(f)^{d} \qquad (2.2.1)$$

where d = total transmission distance (km)

f = carrier frequency (kHz)

$a(f)$ = absorption coefficient (dB/km)

α = spreading factor

In addition, signal experiences a spreading loss, which increases with distance. Spreading loss refers to the energy distributed over an increasingly larger area due to the regular weakening of a sound signal as it spreads outwards from the source [4]. In Figure 2, we've displayed the overall transmission loss that occurs in underwater Acoustic channel over a transmission distance of meters at a signal frequency *f* (kHz) is given by

$$TL = k.10logl + l.10log\, a(f) \qquad (2.2.2)$$

K is a spreading factor (k = 2 for spherical spreading, k = 1 for cylindrical spreading, and k = 1.5 for the so-called practical spreading), and $10\log a(f)$ is the absorption coefficient expressed using Thorp's formula for frequencies above 3 KHz. Gives $a(f)$ in dB/km and f in kHz:

$$10\log a(f) = 0.11 \frac{f^2}{1+f^2} + 44 \frac{f^2}{4100+f^2} + 2.75f^2\, 10^{-4} + 0.003 \qquad (2.2.3)$$

While, for lower frequencies the following formula may be used:

$$10\log a(f) = 0.002 + 0.11\frac{f^2}{1+f^2} + 0.011f^2 \quad (2.2.4)$$

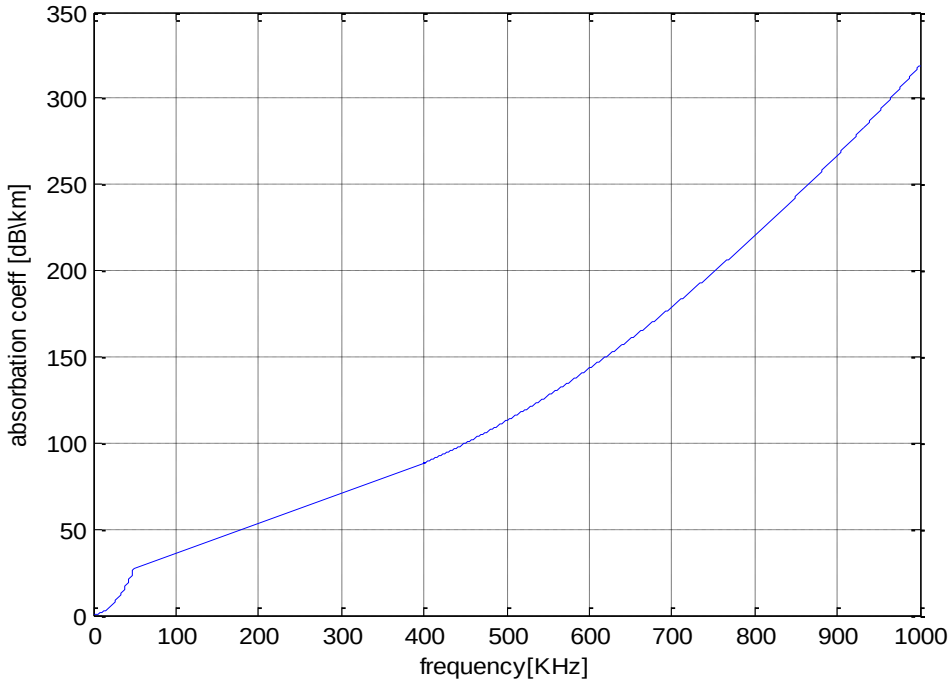

Figure 2**:** Absorption Coefficient

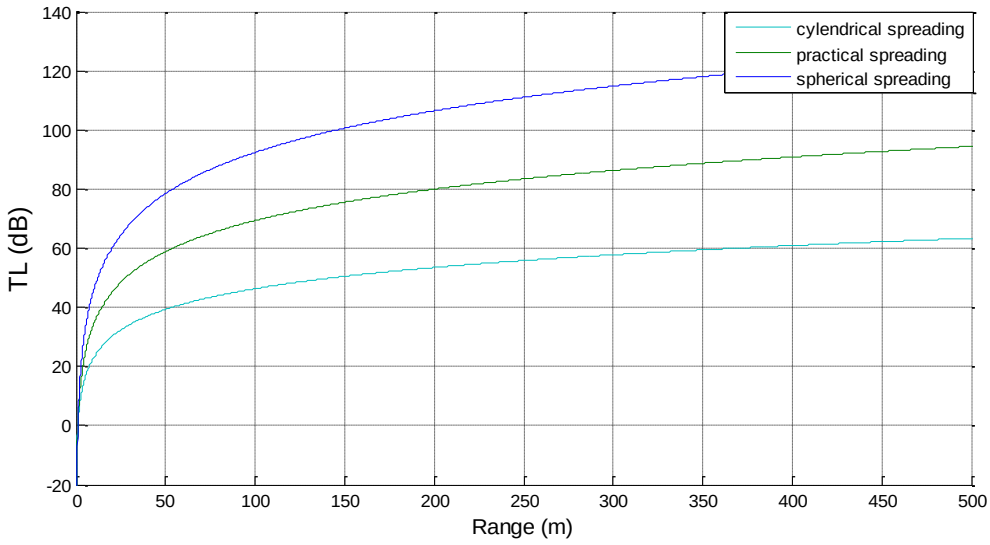

Figure 3: Overall transmission loss (f = 50KHz).

2.3 The Noise Model in UWSNs

The ambient Noise in ocean can be represented as Gaussian, as well as having a continuous power spectrum density (PSD). It consists of four causes (outlined below), where each have a dominating influence in different portions of the frequency spectrum [1].

- Turbulence noise influences only where f < 10Hz

$$10log\ N_t(f)\ =\ 17 -\ 30log\ (f) \qquad (2.3.1)$$

- Shipping noise is dominant only where 10 < f < 100 Hz, and has defined a shipping activity factor of s ranges from 0 to 1 for low high respectively.

$$10log\ N_s(f) = 40\ +\ 20\ (s - 0.5) +\ 26log(f) -\ 60log(f + 0.03) \qquad (2.3.2)$$

- Wave noise caused by wind and rain is a major factor in the frequency region of 100Hz - 100 kHz where wind speed given by m/s.

$$10log\ N_w(f) = 50 + 7.5w\text{^}(1/2) + 20log(f)\ -\ 40log\ (f + 0.4) \qquad (2.3.3)$$

- Thermal noise start effect where f >100 KHz

$$10log\ N_s(f)\ =\ 40 + 20log(f)\ . \qquad (2.3.3)$$

The noise of the acoustic underwater channel under discussion is complex additive colored (frequency dependent) Gaussian noise with zero mean and power spectral density (PSD) $N_o(f)$ that decays linearly on the logarithmic scale in the frequency region 1–20 kHz. For the PSD of the noise of an acoustic channel we use the same approximation as given in [3], [4]:

$$10\log N_o(f) = N_1 - \eta\log(f) \qquad (2.1.3)$$

for the positive constant $N_1 = 50$ dB re μ pa [24] and $\eta = 18$ dB/decade [7]. Obviously, the noise PSD of a narrowband underwater network depends on the carrier frequency but it is flat over the communication band.

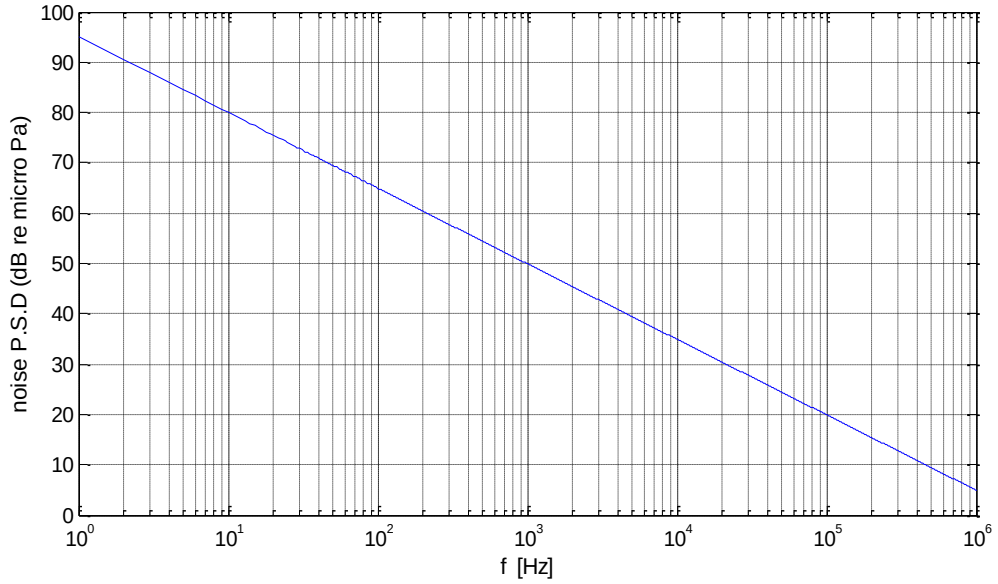

Figure 4: Power spectral density of the ambient noise using the approximation equation in.

2.4 Channel Capacity

The channel capacity strictly depends in transmission distance, carrier frequency, and bandwidth in an underwater acoustic communication. The channel is time variant and the ambient noise is assumed to be Gaussian. Channel capacity is the information rate over the acoustic link which is bounded by the Shannon capacity defined as:

$$C = (BW)\log_2(1 + SNR) \tag{2.3.4}$$

where BW is bandwidth of the system in kHz and SNR indicates the signal to noise ratio.

$$C = BW\, log_2\left(1 + \frac{P}{N}\right) \tag{2.3.5}$$

where P = Power of signal (Watts or Volts^2)

N = Power of noise (Watts or Volts^2)

It can be seen from equation 2.1.5 that, for a source with constant power and limited bandwidth, the capacity of a channel is highly dependent on the noise power. Since, noise and attenuation is a function of distance and frequency, using $A(d, f)$ and PSD $N_o(f)$, the signal-to-noise ratio (SNR) at receiver, at distance d and frequency f for a transmitted power of P is given by [7]:

$$SNR = \frac{P}{BW\ A(d,f)N_o(f)} \tag{2.3.6}$$

Combining equation 2.1.4 and 2.1.6, we get the channel capacity for an acoustic link for optimal link distance and frequency as below [4]:

$$C = (BW)\log_2(1+SNR) = (BW)\log_2\left(1+\frac{P}{(BW)A(d,f)N_o(f)}\right) \tag{2.3.7}$$

2.5 Conclusion

In this chapter some definitions and parameters for the UWSN are reviewed. Moreover, a simulation study shows the effect of the path-loss, noise in narrowband underwater communications channel, and channel capacity is presented.

Chapter 3

Modulation Techniques

In this chapter, we discuss some of the modulation techniques that will give a better performance and optimum information receiving techniques for underwater acoustic communications. In this thesis, we study two different types of modulation and use them in data communication process. PPM is the first one. This modulation technique is used by the vendor (Desert Star Systems). The basic idea of PPM is that the position of each pulse in relation to the position of a recurrent reference pulse is varied by each instantaneous sampled value of the modulating wave. In our experiment, we use m-sequence modulation and Binary Phase Shift Keying (BPSK) modulation, so we focus on these two.

3.1 PN-sequence modulation

Maximal length sequences are generated by using linear feedback shift register. In [12] it shows that shift registers result a binary output that is periodic with period, $N = 2^n - 1$, where n is the degree of a polynomial. For n –stage shift register, the total number of non-zero states will be $2^n - 1$ all having period of, $N = 2^n - 1$ .Shift registers sequence having maximum possible period for a n –stage are called maximal-length sequences or m- sequences. Maximum length sequences are very useful in spread spectrum technology because of following properties:

1. A maximal length sequence contains one more one than zero. The number of ones in the sequence is $\frac{1}{2}\ (N + 1)$.
2. The modulo-2 sum of an m-sequences and any phase shift of the same sequences is another phase of the same m-sequence (shift-and -add property)
3. If a window of width r is slid along the sequence for N-shifts each r-tuple except the all zero r-tuple will appear exactly once.

The periodic autocorrelation function is mathematically expressed as:

$$\theta_b(k) = \begin{cases} 1 & for\ k = lN \\ -\frac{1}{N} & for\ k \neq lN \end{cases} \tag{3.1.1}$$

where, l is any integer and N is the sequence period.

In our thesis, we generate the PN-sequence for the code length of 1024 by choosing the 10^{th} order polynomial.

3.2 Direct Sequence Spread Spectrum (DSSS)/ Binary Phase Shift Keying (BPSK)

The DSSS process is performed by multiplying a carrier frequency and a pseudo-noise (PN) digital signal. The PN code is modulated onto an information signal using modulation techniques binary phase-shift keying (BPSK). For a DSSS system, the spreading rate is increased with higher number of chips per data symbol, which will lead to an improvement of the SNR per symbol in additive white Gaussian noise (AWGN). For time-varying channels, the gain depends on the stability of the channels environment that wave propagates in. The gain of the receiver is reduced if the channel changes considerably during one symbol period. This results in a net loss when the length of the spreading code is increased past a certain point.

Comparison of DSSS and FHSS based on performance depends on the characteristics of an acoustic channel. According to simulation done in [10], the primary limitation is the time variability. The DSSS performs very well when coupled with a chip rate equalizer. The variation in the channel gives the limitation on the performance of the receiver at high spreading rates. The FHSS is more vulnerable to the Doppler shift effect because the signals are transmitted in narrow bands, but more robust to multiple access interference (MAI) than DSSS. FHSS has a higher bit error rate than DSSS, though FHSS has simple receivers and gives robustness to the near-far problem especially when used with convolution coding and soft-decision Viterbi decoding, thus simplifying the power control functionality. However, performance for FHSS is limited in frequency-selective channels.

In DSSS the received signal is multiplied with the original pseudo noise code and integrated over the period on the data symbol, also known as de-spreading. The de-spreading will de-correlate the multiple access interference made by multiple signals. The problem with this method is that it has a very high complexity and can be a problem to implement in small sensor with limited power [3].

Even though there are advantageous and disadvantageous for both direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS) the DSSS has the best performance in underwater acoustic communication. Therefore DSSS is chosen as the spread spectrum technique for the simulation in this study.

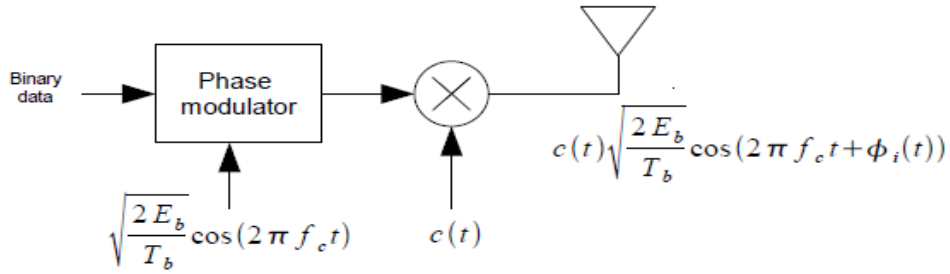

Figure 5: BPSK Transmitting channel model

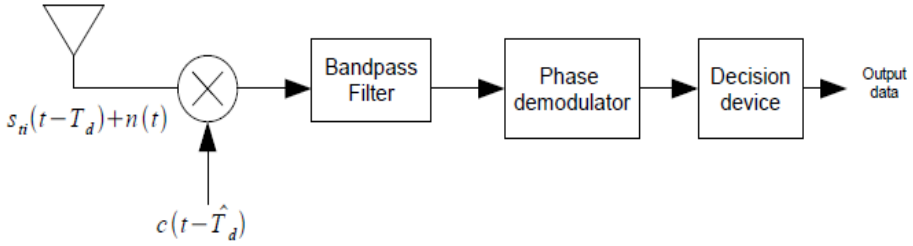

Figure 6: BPSK Receiving channel model

The advantages of direct sequence spread spectrum (DSSS) mentioned above are good arguments for using DSSS in underwater acoustic communication. In addition, its performance in simulations is also quite good and with some optimization the results can be better. The challenge in DSSS is to have good time synchronization between the transmitted code spreader and the received code de-spreader. To reduce the problem of noise, better filters can be used in the receiver. The disadvantage with using spreading codes is that the bandwidth is increased and this can be a problem in band-limited situations. To deal with multiple users and noise, the DSSS is used with Gold codes as spreading codes. These spreading codes work for detecting the correct symbols.

Chapter 4

Information Theoretic Approaches

Here, we review the information theoretic aspects tools and concepts we used in our analysis and evaluation on the performance of the UWSN for cascaded and parallel sensor network. Also, we study the relation between mutual information and channel capacity in relay acoustic sensor network, achieving maximum data rate which is upper bounded by the capacity.

Entropy, joint and Conditional entropy

4.1.1 Entropy

The most fundamental concept of information theory is the entropy. The entropy of random variable X interpreted as the expected value of $-\log_2 p(X)$ where X is drawn according to the probability mass function $p(x)$ [6]:

$$H(x) = E_p \log_2 \frac{1}{p(X)} = \sum_{x_i \in X} p(x_i) \log_2 \left(\frac{1}{p(x_i)}\right) \qquad (4.1.1)$$

where $p(x_i)$ is probability distribution of x_i.

We measure the entropy of transmitted and received message signal, where we use the PN-sequence as transmitted and received signals, which will be a binary sequence due to PPM modulation used by default in acoustic modem set.

4.1.2 Joint Entropy

The joint entropy measures how much uncertainty there is in the two random variables X and Y taken together. The joint entropy is given by

$$H(X,Y) = -\sum_{x,y} p(x,y) \log p(x,y) \qquad (4.1.2)$$

4.1.3 Conditional Entropy

The conditional entropy is a measure of how much uncertainty remains about the random variable *X* when we know the value of *Y*. The conditional entropy of X given Y is

$$H(X|Y) = \sum_y P_Y(y) \left[-\sum_x P_{X|Y}(x|y) \log(P_{X|Y}(x|y)) \right] \quad (4.1.3)$$

4.2 Mutual information

The mutual information *I(X; Y)* is the relative entropy between the joint distribution and the product distribution [20].

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (4.2.1)$$

Mutual information is also known as the reduction of entropy of *X* when *Y* is known. It is symmetric in the arguments. That is, I (X; Y) = I (Y; X) as well as it is non-negative value as it follows from the definition of entropy and mutual information:

$$I(X;Y) = H(X) - H(X|Y)$$

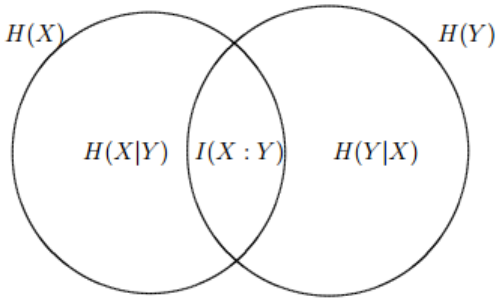

Figure 7: Graphical representation of Conditional Entropy and MI of X and Y

Information Loss

Information loss *(L)* between two different sensors network models is calculated from conditional entropy of input and output data [6].

$$L(X \longrightarrow Y) = H(X|Y)$$

$$H(X|Y) = \sum_y P_Y(y) \left[-\sum_x P_{X|Y}(x|y)\log(P_{X|Y}(x|y))\right] \quad (4.2.2)$$

Similarly,

$$H(Y|X) = \sum_{x_i \in X} p(x_i)\, H(Y|X = x) = \sum_{x_i \in X} p(x_i)\, H = H(p) \quad (4.2.3)$$

Similarly, to calculate the information loss in between the cascaded channel system, we individually estimate the conditional entropy for each channel communication.

Intuitively, in cascaded channel, output Y of input data X is directly used as input to transmit to Z .This gives us:

$$\boldsymbol{L(X \rightarrow Z) \geq L(X \rightarrow Y) + L(Y \rightarrow Z)}$$

4.3 Bit Error Rate (BER)

Bit error rate, BER is a key parameter that is used in assessing systems that transmit digital data from one location to another. Systems for which bit error rate, BER is applicable include radio data links as well as fiber optic data systems, Ethernet, or any system that transmits data over a network of some form where noise, interference, and phase jitter may cause degradation of the digital signal[21].

$$BER = \frac{Number\ of\ erros}{Total\ number\ of\ bits\ sent} \quad (4.3.1)$$

As the name implies, a bit error rate is defined as the rate at which errors occur in a transmission system. This can be directly translated into the number of errors that occur in a string of a stated number of bits.

BER and Eb/No

Signal to noise ratios and Eb/No are parameters that are more associated with radio links and radio communications systems. In terms of this, the bit error rate, BER, can also be defined in terms of the probability of error or POE. The determine this, three other variables are used. They are the error function, erf, the energy in one bit, Eb, and the noise power spectral density (which is the noise power in a 1 Hz bandwidth), No.

It should be noted that each different type of modulation has its own value for the error function. This is because each type of modulation performs differently in the presence of noise. In particular, higher order modulation schemes (e.g. 64QAM, etc) that are able to carry higher data rates are not as robust in the presence of noise. Lower order modulation formats (e.g. BPSK, QPSK, etc.) offer lower data rates but are more robust.

Factors affecting BER

It can be seen from using Eb/No, that the bit error rate, BER can be affected by a number of factors:

- Interference: The interference levels present in a system are generally set by external factors and cannot be changed by the system design. However it is possible to set the bandwidth of the system. By reducing the bandwidth the level of interference can be reduced. However reducing the bandwidth limits the data throughput that can be achieved.
- Increase transmitter power: It is also possible to increase the power level of the system so that the power per bit is increased. This has to be balanced against factors including the interference levels to other users and the impact of increasing the power output on the size of the power amplifier and overall power consumption and battery life, etc.
- Lower order modulation: Lower order modulation schemes can be used, but this is at the expense of data throughput.
- Reduce bandwidth: Another approach that can be adopted to reduce the bit error rate is to reduce the bandwidth. Lower levels of noise will be received and therefore the signal to noise ratio will improve. Again this results in a reduction of the data throughput attainable.

4.4 Optimum Channel Capacity

According to Shannon (1948), channel capacity is defined as channel's mutual information maximized over all the possible input distribution. Channel capacity is an upper bound of the mutual information that gives the maximum data transmission rate that can be reliably transmitted through a communications channel. The information channel capacity of a discrete memoryless channel is [20]:

$$C = max_{p(x)}\, I(X;Y) \qquad (4.1.1)$$

This quantity is defined as the maximum of the average mutual information $I(X; Y)$ between input information X and its corresponding output information Y of the channel over a choice of distribution of X. Mutual information satisfies $I(X;\ Y\ ) =\ I(Y\ ;X) \geq 0$.

4.5 Mutual Information over Cross correlation

Mutual information is an ideal metric for our sensor placement problem in multi-hop because it results in minimizing the conditional entropy of our object state. In long range underwater communication, placement of multiple sensors in hop increases the mutual information among transmitter and receiver reducing bit error rate, information loss, and large propagation delay.

Mutual information identifies and quantitatively characterizes relationship between data sets that are not detected by commonly used linear measures of correlation. Given two data sets as input and output of channel, $\{X\} = \{x_1, x_2\ldots, x_n\}$ and $\{Y\} = \{y_1, y_2, \ldots. y_N\}$, their mutual information, $I(X,Y)$, is the average number of bits of $\{X\}$ that can be predicted by measuring $\{Y\}$. Mutual information relationship is symmetrical, $I(X, Y) = I(Y, X)$. Analyzing observational data, calculation of the mutual information occurs in two contexts:

i) Identification of nonlinear correlation,

ii) In the investigation of causal relationships with directed mutual information.

Calculation of mutual information is important in the characterization of causal relationships between two information data set. Mutual information captures complex association between variables that are missed by standard correlation measures. By definition, a correlation measures, either linear or nonlinear, quantifies the degree of correlation between $\{X\}$ and $\{Y\}$ under their respective definitions, but it does not necessarily identify causal relationships in the sense of identifying which variable drives the other, if indeed such a relationship exists. However, the principle of the mutual information maximization follows naturally from the expected uncertainty minimization criterion in a Bayesian filtering framework [22].

4.6 Relating Information Theory to our Network Model

4.6.1 Collaborative Sensor Network system

In UWC, acoustic links and their propagation characteristic differ in terms of distance, time, and multipath spreads. Collaborative sensor network model influences the acoustic communication in order to state the challenges posed by path loss, geometric spreading, multi-path, and Doppler spread.

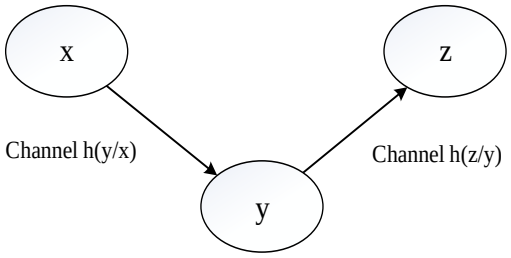

Figure: 8. Cascaded channel model.

The Fig 1 shows that the use of multiple sensor nodes in underwater to form a collaborative network system can increase the information contained in signal Y about signal X[3]. We use the data processing inequality to show that information loss can be reduced forming the sensor nodes as a Markov chain. For data transmitted through $X \rightarrow Y \rightarrow Z$, the conditional distribution of Z depends only on Y and is conditionally independent of X. Specifically the join probability function of for X, Y and Z can be written as:

$$p(x, y, z) = p(x)p(y|x)p(z|y) \qquad (4.6.1)$$

Therefore, the mutual information between random variable X and Z will be improved with addition of another sensor node Y in between the propagating distance of X and Z. This approach of creating n cascaded channels to transmit data from one node to another will effect in the information loss, mutual information, and channel capacity of the system.

Each sensor node in the network becomes a primary transmitter with its own probability density function and new energy level. The objective of our work is to organize the sensors in cascade system to maximize the transmitted signal power along large distance of propagation in an underwater communication [4].

In this sensors network model, we wish to retransmit the received signal to another sensor node exactly as they are received. Thus, in order to obtain the input-output statistics of the cascade, we need only to consider the signal as one sample at a time. Let the sample X of the first transmitter has its corresponding probability p(x). The signal X will transmit down the first channel and will be received as Y1 by the first receiver sensor node, Y2 by the second receiver sensor node, and finally as Yi by the last receiver sensor node. During the observation of downstream transmission of signal data from X to Yi , each channel will have different conditional probability distribution since the channel state is changing with time i.e. for the ith channel $p(y_i|y_{i-1})$ gives the probability distribution of the samples Y_i .

In terms of distance of propagation and energy of wireless sensor node in an underwater, the use of multiple nodes forming a cascade channel will help to reduce the information loss and increase the signal to noise ratio (SNR), thus increasing the channel capacity of the signal [5]. Channel capacity for discrete time Gaussian channel with signal power 'S' is given by,

$$C = \frac{1}{2}\log_2\left(1+\frac{S}{N}\right) \qquad (4.6.2)$$

However, for bandwidth limited performance,

$$C = B\ \log_2\left(1+\frac{S}{BNo}\right) \qquad (4.6.3)$$

For $X \to Y$, we know the channel capacity is equal to

$$C = max_{p(x)}\, I(X;Y) \qquad (4.6.4)$$

However, by cascading and using data processing inequality we have, for $X \to Y \to U \to V$,

$$I(X;V) \le I(U;V) \qquad (4.6.5)$$

We can find the $max_{\ p(x)}$ of both sides,

$$C = max_{p(x)}\, I(X;V) \le C = max_{p(x)}\, I(U;V) = C\left(p_{(x)}\right) \qquad (4.6.6)$$

To find the capacity of the product channel of Markov chain [4] we will obtain:

$$Y_1 \to X_1 \to X_2 \to Y_2$$

$$I(X_1,X_2;Y_1,Y_2) = H(Y_1,Y_2) - H(Y_1,Y_2|X_1,X_2)$$

$$= H(Y_1,Y_2) - H(Y_1|X_1,X_2) - H(Y_2|X_1,X_2)$$

$$= H(Y_1,Y_2) - H(Y_1|X_1) - H(Y_2|X_2)$$

$$\le H(Y_1) + H(Y_2) - H(Y_1|X_1) - H(Y_2|X_2)$$

$$I(X_1,X_2;Y_1,Y_2) = I(X_1;Y_1) + I(X_2;Y_2) \qquad (4.6.7)$$

Therefore,

$$C = max_{p(x1,x2)} I(X_1,X_2;Y_1,Y_2)$$

$$\le max_{p(x1,x2)} I(X_1;Y_1) - max_{p(x1,x2)} I(X_2,Y_2)$$

$$= max_{p(x1)} I(X_1;Y_1) + max_{p(x2)} I(X_2;Y_2)$$

$$C = C_1 + C_2 \qquad (4.6.8)$$

Similarly, to calculate the information loss in between the cascaded channel system, we individually estimate the conditional entropy for each channel communication.

Intuitively, in cascaded channel, output Y of input data X is directly used as input to transmit to Z .This will show us

$$L(X \to Z) \geq L(X \to Y) + L(Y \to Z) \qquad (4.6.9)$$

4.6.2 Parallel Sensor Network system

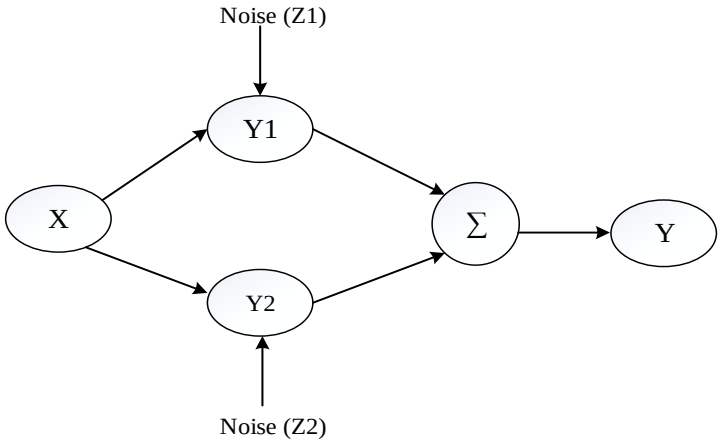

Figure 9: Parallel sensor network system

Let us consider the power of signal X be P. Assuming additive white Gaussian noise, Z1 and Z2 with zero mean and variance σ^2,

$$Var(Z1 + Z2) = E[(Z1 + Z2)^2] \qquad (4.6.10)$$

The channel can be reduced to,

$$2X + (Z1 + Z2) = Y \qquad (4.2.11)$$

By using the equation (14) and (15), the channel capacity formula is,

$$C = \frac{1}{2}\log_2\left(1 + \frac{2P}{2\sigma^2}\right) \qquad (4.2.12)$$

When we have cost constraint on the power, we need to optimize the total capacity of the two parallel channels,

$$C = \frac{1}{2}\log_2\left(1 + \frac{P_1}{N_1}\right) + \frac{1}{2}\log_2\left(1 + \frac{P_2}{N_2}\right) \qquad (4.2.13)$$

Let the two outputs Y1 and Y2 be conditionally independent and conditionally identically distributed given X. Thus,

$$p(y_1, y_2|x) = p(y_1|x)p(y_2|x) \qquad (4.2.14)$$

Therefore,

$$\begin{aligned} I(X; Y_1, Y_2) &= H(Y_1, Y_2) - H(Y_1, Y_2|X) \\ I(X; Y_1, Y_2) &= H(Y_1) + H(Y_2) - I(Y_1, Y_2) - H(Y_1|X) - H(Y_2|X) \\ I(X; Y_1, Y_2) &= I(X, Y_1) + I(X, Y_2) - I(Y_1, Y_2) \\ I(X; Y_1, Y_2) &= 2I(X, Y_1) - I(Y_1, Y_2) \qquad (4.2.15) \end{aligned}$$

Hence, the capacity of channel $X \rightarrow (Y_1, Y_2)$ is

$$\begin{aligned} C_2 &= max_{p(x)} I(X; Y_1, Y_2) \\ &= max_{p(x)} 2I(X; Y_1) - I(Y_1, Y_2) \\ &\leq max_{p(x)} 2I(X; Y_1) \\ C_2 &= 2C_1 \qquad (4.2.16) \end{aligned}$$

Similarly, information loss in our discrete data communication can be calculated from conditional entropy estimation for between input and output data. Comparing the best output result from parallel channel, we estimate the conditional entropy to obtain the information loss rate [6].

$$L(X \longrightarrow Y) = H(X|Y)$$

$$H(X|Y) = \sum_y P_Y(y) \left[-\sum_x P_{X|Y}(x|y) \log(P_{X|Y}(x|y)) \right]$$

where $P_Y(y)$ is marginal pmf for output Y and $P_{X|Y}(x|y)$ is the channel error estimation.

In parallel channel network system,

$$L(X \rightarrow Y_1) \geq L(X \rightarrow Y)$$
$$L(X \rightarrow Y_2) \geq L(X \rightarrow Y)$$

4.7 Conclusion

In this chapter, information theoretic tools like entropy, channel capacity, mutual information, joint entropy, and conditional entropy are discussed. Also, we relate our sensor network models to the information theoretic aspects, as well as showing the reason of choosing mutual information over correlation to study the performance of network model.

Chapter 5

Experimental Approach

After a preliminary research and signal analysis, we concentrate our study to the main objectives of our thesis: a) analyzing and studying the impact of using mobile/moving sensor nodes, as opposed to fixed nodes in cascaded and parallel UWSN, b) An experimental approach to enhance the performance of UWSN.

5.1 Hardware and Software Details

5.1.1 Equipment

Our experimental model consisted of an indoor swimming pool, two pairs of SAM-1 acoustic transducers provided by Desert Star Systems, four portable computers all of them equipped with MATLAB software, signal processing toolbox, and communication toolbox, a hydrophone to measure sound underwater, and an acoustic speaker to generate noise.

The modems can operate in the range of 250 meters. The transmitting power of sensors depends on the voltage supply (ranging from 183dB at 8V to 189dB at 16V). The acoustic modems are set to transfer serial data at 4800 baud, 8 data bits, no parity, and 1 stop bit. Flow control should be set to software handshaking. However, we observe that software handshaking would affect data rate. The modems were configurable to a specific data rate. However, increase in data rate decreased the range of transmission. Therefore, we configured the sensors to operate in 13 bits per second. Details of the hardware are shown in the Table 2.

Figure 10: Pair of the Acoustic Modem used in experiments

TABLE 2: Specification of the modem

Type of command /specification	Operating function/condition of sensors
###	Sets to a command mode for configuration.
We used S5 for transmission @ 13bits /sec	Set acoustic transmit data speed
We used R5 for receiving @13bits/sec	Set acoustic receive data speed
Serial buffer	32 byte
Operating temperature	0-70 ^{0}C
Filter type	4th order band pass filter
Operating frequency for single channel receiver	33.8khz
Operating frequency of modem	33khz-42khz Omnidirectional
Sonar Data format	16- bit data + 4 bit checksum per word
Modulation	Pulse Position Modulation

5.1.2 Software development (SAM Control)

SAM Control is a GUI program that is written for these experiments. The program is written in MATLAB, which reads and writes data to the serial buffer of the modem. SAM Control greatly streamlines the data collection and data analysis process by automating commonly used functions of the modem. Multiple instances of the program can be run so that one computer can control multiple modems simultaneously. Its main features:

• COM settings: Configure the COM port settings that are connected to the modem.

• Modem Settings: These sets of control are used to configure the sensor communication parameters, such as receiver threshold, transmission speed (data rate) and receive speed.

• TX Mode: mainly responsible of generating the transmitting data which are the m-sequences of 1023 length so to be send through the sensor.

• RX Mode: managing all operations need to be done by the receiver, including reading the received data and storing it in a desired file type, so it can be ready for performing our analysis and evaluation on the data.

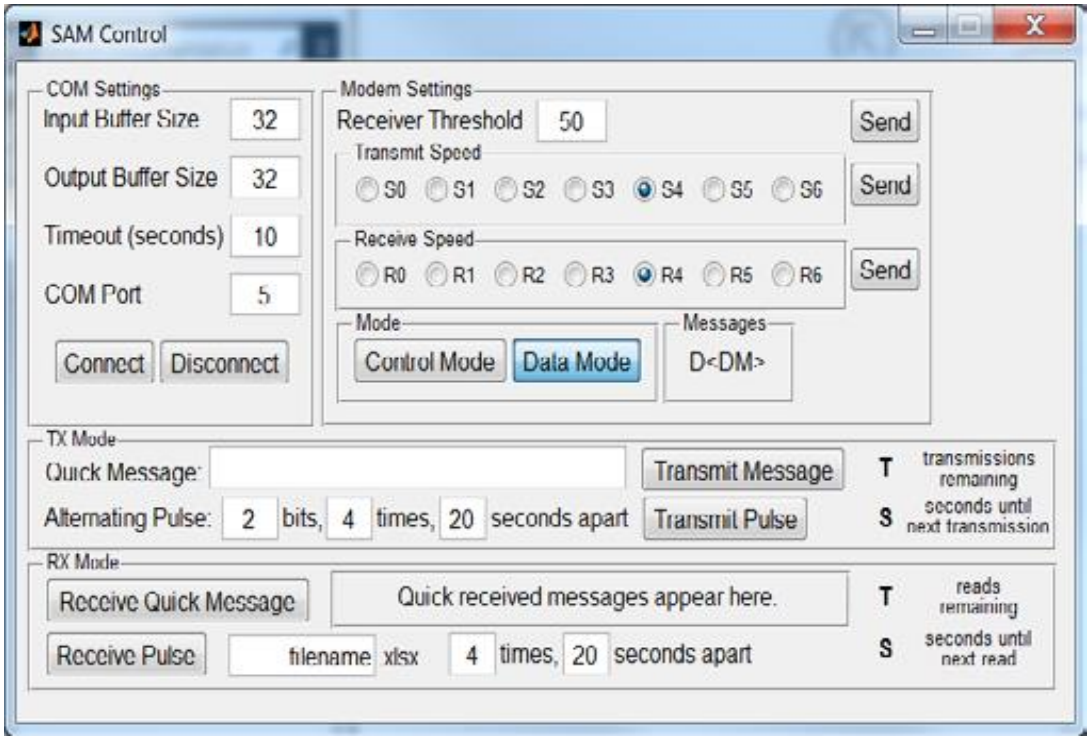

Figure 11: SAM Control GUI

During our experiment, we configured the sensors for serial communication, generated PN-sequence of the code length of 1024 by choosing the 10th order polynomial. We chose 10th order polynomial so that we could get the sequence with period length of 1023bits. We transmitted baseband m-sequences of length 1023. Initially, we were not able to receive 1023 bits because of 32 bytes buffer memory of the sensors used. The buffer memory size was only 32 bytes and each bit sent via serial port was codded as 8 bits. As a result, we were only able to send maximum of 30 bits plus 2 bytes handshaking at a time from the m-sequences of length 1023. Therefore, practically the length of the sequence could be considered 30 for one transmission.

5.2 Collaborative network using fixed sensor nodes.

In cascaded sensor network we transmitted data from sensor 1 and received the output data in sensor 2. Then we transmit the received data from sensor 2 exactly as it was received to sensor 3. And same process was repeated to send data from sensor 3 to sensor 4. The output at sensor 4 was compared with input data at sensor 3 and then we calculated mutual information. Besides, bit error rate, probability error, and information loss is calculated at each communication channel. The results obtained are presented in next Section of experimental results.

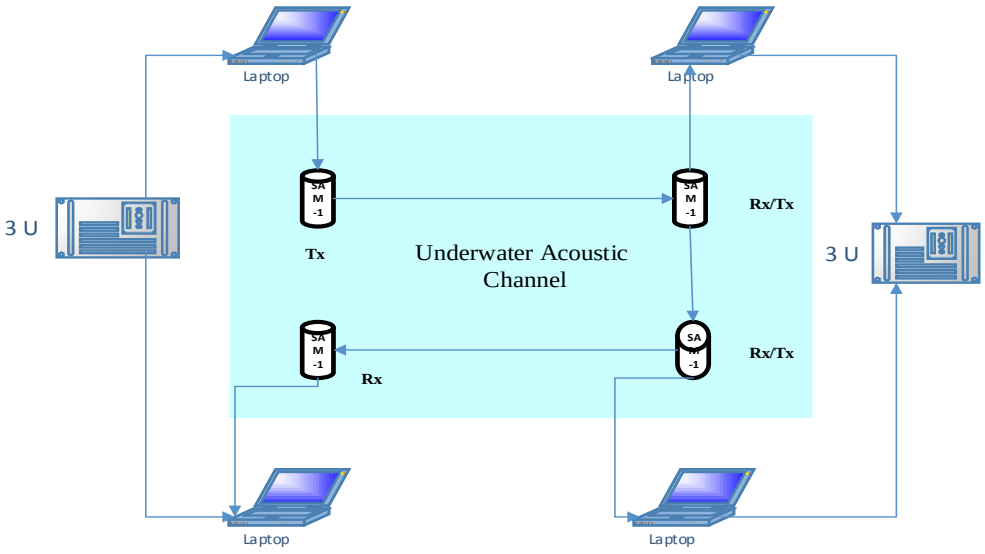

Figure 12: Cascaded underwater wireless sensor network

Collaborative Network Result.

We obtained the experimental result for cascaded wireless sensor network in an underwater acoustic communication for four sensors. The bit loss and bit error were computed in an underwater channel sending 1023 bits of 1's and 0's alternately for couple of times for different location of sensors.

Table 3: Bit loss test for different distance of sensors

Distance between sensors	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
5 m	1023	1015	8	0.007
7 m	1023	1011	12	0.011
10 m	1023	1011	12	0.011

In Table 3, we transmitted 1023 bits of binary sequences data to the receiver sensor node at a distance of 5 m. This channel transmitted all 1023 bits; however it has bit error of 8 bits in average.

Table 4: Result analysis for Tx1 and Rx1 (d=5 m)

Bits transmitted	Bits received	BER	*I(X;Y)*	Information loss *H(X|Y)*
1023	1023	0.0078	1.00	0.0
1023	1023	0.0087	1.00	0.00
1023	1023	0.0078	1.00	0.00

In Table 4, the transmitted data were exactly the received data from first senor node; however we filtered the error bits that were incurred in first channel. This channel was also set for distance of 5 m between transmitter and receiver. We saw a full communication of data between sensor nodes with bit error of 7 -8 bits in average.

Table 5: Result analysis for Tx2 and Rx2 (d=5 m)

Bits transmitted	Bits received	BER	*I(X;Y)*	Information loss *H(X|Y)*
1015	1015	0.0068	1	0
1014	1014	0.0078	1	0
1015	1015	0.0088	1	0

In Table 5, we load the received data from sensor node 3 and transmitted exactly to another sensor node in cascaded network. This receiver was at a distance of 5 m. The channel had full communication of data with bit error of approximately 7-8 bits.

Table 6: Result analysis for Tx3 and Rx3 (d=5 m)

Bits transmitted	Bits received	BER	*I(X;Y)*	Information loss *H(X|Y)*
1008	1008	0.0079	1	0
1007	1007	0.0079	1	0
1006	1006	0.0089	1	0

Table 6 shows our improvement regarding drop in bit loss and bit error compare to direct transmission between two senor nodes and cascaded network. The channel distance was approx. 15 m. The output result shows that the channel has bit error along with bit loss. This proves that transmission loss in underwater acoustic communication depends on distance.

Table 7: Result analysis for Tx1 and Rx4 (d=15 m)

Bits transmitted	Bits received	BER	*I(X;Y)*	Information loss *H(X|Y)*
1023	1012	0.0185	0.9156	0.0844
1023	1015	0.0156	0.9419	0.0581
1023	1009	0.0215	0.8952	0.1048

Table 7 shows that underwater acoustic communication (UAC) capacity is strongly dependent on transmission distance. Using more than one sensor in cascaded network system will increase in the mutual information and rate of data transmission with minimum bit error probability.

Table 8: Result analysis for cascaded sensors network

S.N	Case study	*I(X:Y)*	BER	Information loss *H(X|Y)*
1	Tx_1 and Rx_1	1	0.0081	0
2	Tx_2 and Rx_2	1	0.0078	0
3	Tx_3 and Rx_3	1	0.0082	0
5	Tx_1 and Rx_4	0.9054	0.1853	0.0824

5.3 Parallel network using fixed sensor nodes.

A similar experiment was performed in parallel sensor network system. In this system, we have one transmitter and 3 receiver, acting system as Single Input Multiple Output (SIMO) system. The optimum output was recorded each time and the experiment was carried out several times for *m* sequences of length 1023 bits. The result obtained for probability of error, BER, mutual information, entropy estimation is displayed in numerical results of Section IV.

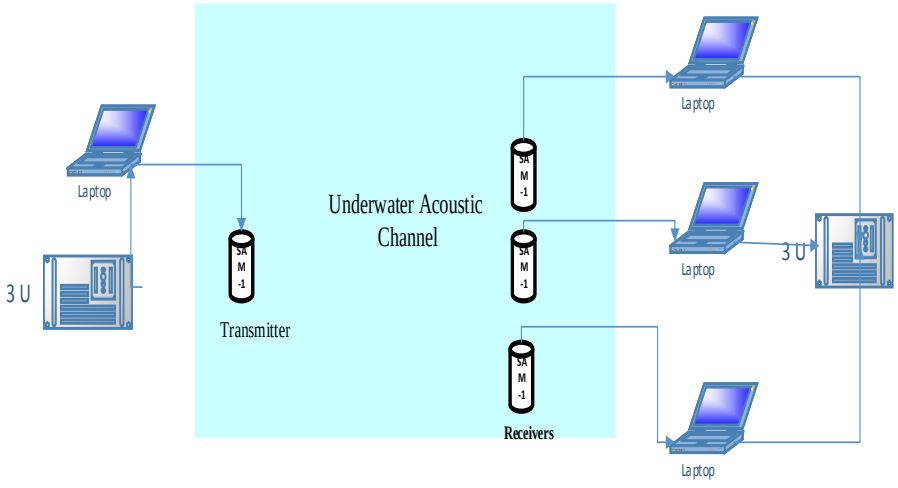

Figure. 13. Parallel underwater wireless sensor network

Parallel network results.

Similarly, an experimental result for parallel sensor network model shows the following outcomes shown in Table 9. This channel network is more effective than cascaded sensors network channel; however cascaded network can transmit data to larger distance for available sensor nodes. In parallel sensor network channel, bit error rate is significantly low compared to cascaded sensor networks. This tells us that data transmission in parallel sensor network has low probability error. The bit loss and bit error in data transmission between X and Y_1 might be fixed during data transmission with second or third sensor node.

Table 9: Result analysis for Tx1 and Rx1

Bits transmitted	Bits received	BER	$I(X;Y)$	Information loss $H(X\|Y)$
1023	1019	0.0039	0.9804	0.0196
1023	1016	0.0068	0.9645	0.0355
1023	1017	0.0058	0.9700	0.0300

The maximum distances between two communicating sensors were 15 meters. The sensors were positioned such that they could float in water and move freely. We are optimizing the sensors locations and sensitivity analysis to increase the mutual information and channel capacity [7].

5.4 UWSN network using moving sensor nodes

In the experimental model, we studied the impact of using moving acoustic sensor node in a point to point communication system to accomplish same work done in the cascaded network model above.

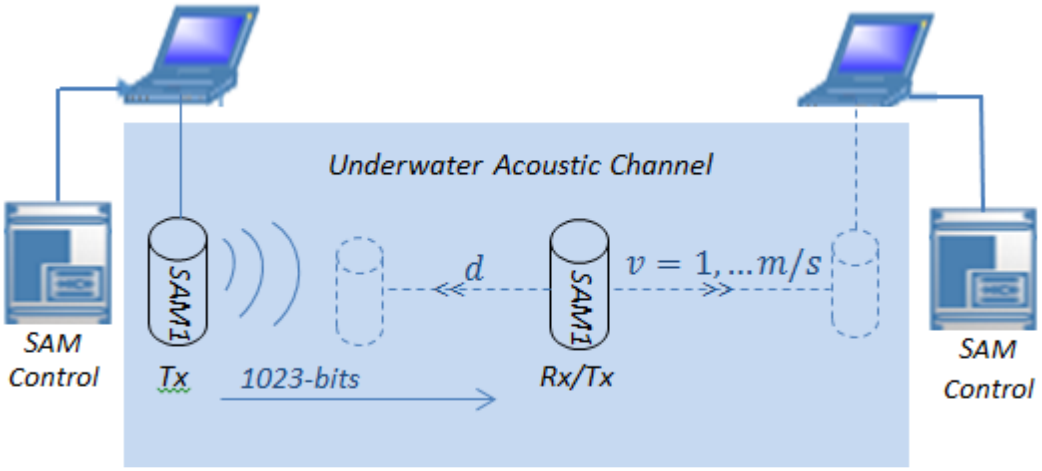

Figure 14: Underwater Mobile Acoustic Sensor Network

The receiver has been moving while receiving the transmitted data in both vertical and horizontal directions, the maximum distances reached between the two sensor nodes is 15m, as we limited it by the size of the indoor pool, where the experiments took place. We have done the transmission of the data several times over different distances with different velocities. All the experimental observations and result we obtained for probability of error, BER, mutual information, entropy estimation is displayed in numerical results in section below.

UWSN network using moving sensor nodes results

In table 9, the bit loss and bit error were computed in an underwater channel sending 1023 bits of 1's and 0's alternately for a couple of times for different velocities of the moving sensor over 15m maximum distance apart between the sensors.

In table 10, the bit loss and bit error were computed in an underwater channel sending 1023 bits of 1's and 0's alternately for a couple of times for different velocities of the moving sensor over 15m maximum distance apart between the sensors.

Table 10: Bit loss test for mobile sensor nodes of maximum distances = 15m.

Sensor Velocity	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1 Knot	1023	1018	5	0.004
2 knots	1023	1018	5	0.004
3 knots	1023	1015	8	0.007

Table 11: Bit loss test for mobile sensor nodes moving in vertical direction.

Sensor Velocity	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1 Knot	1023	1018	5	0.004
2 knots	1023	1018	5	0.004
3 knots	1023	1015	8	0.007

In the table below, we show the information theoretic aspects, we use to evaluate the performance of our point to point communication system model using mobile sensors nodes moving with velocities = 1 knot, 2knots, 3knots.

Table 12: Performance results of using mobile sensor nodes

V	Data length	Bits received	P (E)	*I(X;Y)*	Information loss *H(X|Y)*
$v1$	1023	1023	0.004	1.00	0.0
$v2$	1023	1023	0.004	1.00	0.00
$v3$	1023	1023	0.007	1.00	0.00

X and Y are the transmitter and receiver nodes respectively in our communication system. As shown in Table 4, we achieve maximum mutual information (I(X, Y) = 1), since we have received

same sequence length of the transmitted data, However, there is a small probability of error in reconstructing the transmitted bits. In conclusion, after estimating a simulation study of the UAC model, we used the information theory aspects tools to compare the performance of our UWSN model using mobile sensor nodes to the UWSN model that the author proposed [3].

We observed remarkable results in terms of the performance based on probability of error, mutual information and loss of information. The result of this work indicates that the use of few mobile sensors in UWSN can accomplish the same work as a larger group of static networks and achieve comparable results with low probability of error ≈0.001.

5.5 Experiment Approach based on the BER.

In this section, we demonstrate our experimental approach to enhance the power consumption and the performance of acoustic sensors network.

5.5.1 Method

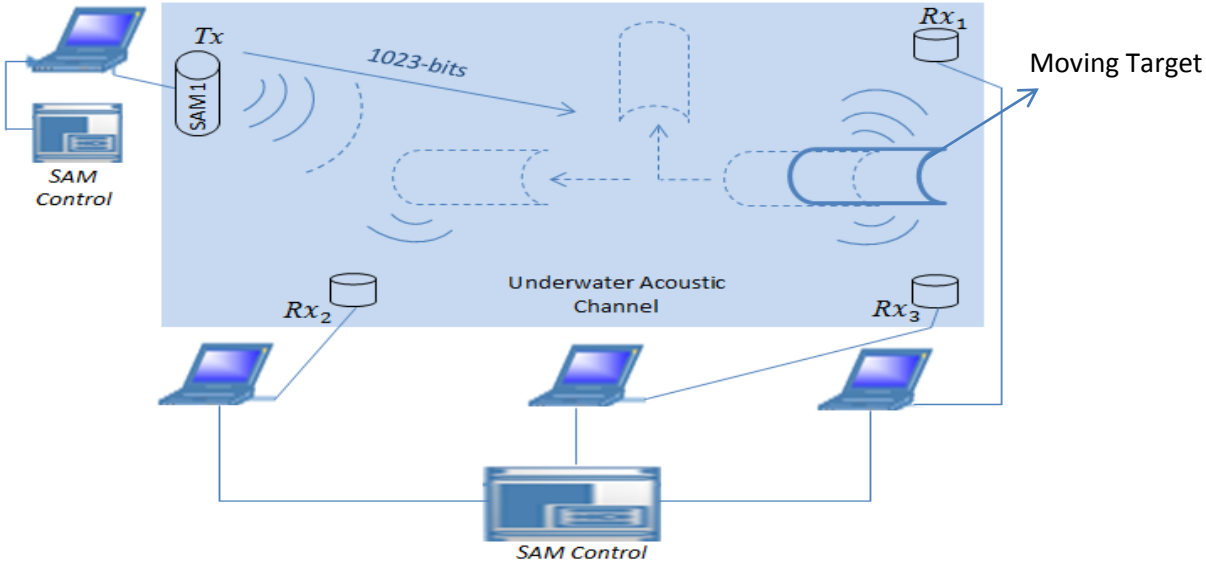

Figure 15: Experimental approach setup

In the distributed sensor network model shown in fig.15, we have one transmitter and 3 receivers optimally distributed, acting system as a Single Input Multiple Output (SIMO) system. In this system, we started by transmitting the data while only one operating receiver (sensor) and the

rest of the sensors were not functioning. The optimum output was recorded each time for m-sequences of length 1023 bits. Our developed software is continuously analyzing the received data and based on the results calculations of the loss of information, BER and mutual information, it sends a signal to operate other sensors. While collecting the data from multiple sensors/receivers, the software process compares and combines the data to achieve the desired BER value and signal detection quality with the least number of operated receivers. The user can set a threshold for the BER and the MI, so it compares it to the results and maintain that desired threshold. Several Experiments have been done in school's indoor pool over different environments (existence and nonexistence of moving targets) to verify our approach and results.

5.5.2 Scenario I

In table 13, the bit loss and bit error were computed for scenario 1 where no moving target or in clear underwater channel sending 1023 bits of 1's and 0's alternately for a couple of times with no effect of noise by moving target or high disturbance waves. The maximum distances were 15m between the sensors. We set the threshold BER = 0.003. It means that the least number of sensors will be used to maintain signal quality or the BER for the transmitted sequences no more than 0.003 (3bits error out of every 1000 transmitted bits) and since The BER less than desired value, only one sensor is functioning and collecting data while the rest of the deployed sensors are not functioning and result in reducing power consumption by the network.

Table 13: Numerical results for scenario #1

No. of operated sensors	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1	1023	1021	2	0.0019
1	1023	1021	2	0.0019
1	1023	1022	1	0.0009

5.5.3 Scenario 2: Target moving (External noise)

In the table below, we show the analysis and results for scenario2 where there is a target moving and making noise. During the transmission we used a moving boat to have a similar effect of moving object as well as increase the noise level in the underwater channel while the BER threshold was set to 0.003.

Table 14: Numerical results for scenario #2

No. of operated sensors	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1	1023	1011	12	0.01
2	1023	1017	6	0.005
3	1023	1015	2	0.001

As shown in table 14, the BER of the transmitted data decreasing regularly. Since, the BER for the first sequence transmitted data is less than the threshold, the software sends signal to turn on the next adjacent sensor to start recording data and as a result we can see the BER decreased to 0.005. But, since the BER still less than the desired value (0.003), it turns another sensor to collect the data from the three sensors and by combining and selecting the data with lowest probability of error, so that it achieves a better signal detection and BER.

Chapter 6

Discussion, Results and Conclusion

In this section, we show summary of the measurements and a comparison of the numerical results on the performance of the UWSN using fixed and moving acoustic sensor nodes.

Before starting our experimental observations, we used the hydrophone to practically find the PSD of the noise in UAC and its relation with the transmitted frequency. The transmitted signal has a carrier frequency of 17 kHz and the sampling rate of 8000. From the Figure 16, we can see that the power of noise is high for frequency range of 0 to 5 kHz. As the frequency increases we can observe that the power of noise is less than 38dB. So if we are able to transmit the signal in the frequency range where the noise power is relatively less, we can obtain better performance. Therefore, we choose 17 kHz of carrier frequency for our experiments.

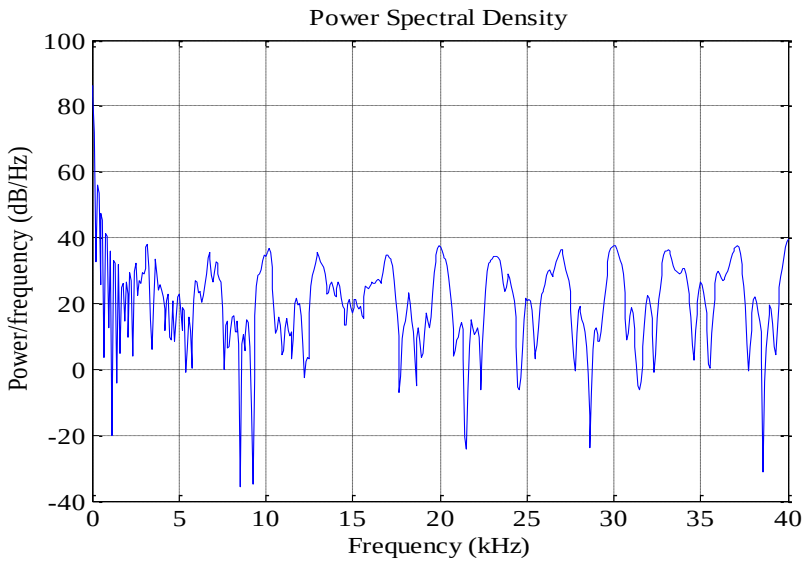

Figure:16. PSD of Noise (N_o (f)) experimentally measured in underwater

Table 15: Result analysis of using fixed Sensor nodes

S.N	Case study	*I(X:Y)*	BER	Information loss *H(X|Y)*
1	Tx_1 and Rx_1	1	0.0081	0
2	Tx_2 and Rx_2	1	0.0078	0
3	Tx_3 and Rx_3	1	0.0082	0
5	Tx_1 and Rx_4	0.9054	0.1853	0.0824

6.2 Conclusion

We used the information theory aspects tools to compare the performance of our UWSN model using mobile sensor, as opposed to fixed sensor nodes in UWSN. We observed remarkable results in terms of the performance based on the probability of error, mutual information and loss of information. Our result indicates that the use of few mobile sensors in UWSN can accomplish the same work as a larger group of static networks and achieve comparable results with low probability of error ≈ 0.001.

Table 16: Result analysis of using fixed sensor nodes

S.N	Case study	*I(X:Y)*	BER	Information loss *H(X|Y)*
1	Tx_1 and Rx_1	1	0.0081	0
2	Tx_2 and Rx_2	1	0.0078	0
3	Tx_3 and Rx_3	1	0.0082	0
5	Tx_1 and Rx_4	0.9054	0.1853	0.0824

Table 17: Summary results for using mobile sensor nodes

V	Data length	Bits received	P (E)	*I(X;Y)*	Information loss *H(X|Y)*
$v1$	1023	1023	0.004	1.00	0.0
$v2$	1023	1023	0.004	1.00	0.00
$v3$	1023	1023	0.007	1.00	0.00

Also, another significant contribution is we show the results of applying our experimental approach to UWSN can reduce the BER of the communications system while at the same time maintaining low power consumption depending on the underwater environment.

Table 18 (no target is moving): Numerical results for scenario #1

No. of operated sensors	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1	1023	1021	2	0.0019
1	1023	1021	2	0.0019
1	1023	1022	1	0.0009

Table 19: Approach results in Noisy UAC (Scenario#2)

No. of operated sensors	No. of bits transmitted	Received no. of bits	Bit loss	Bit error rate
1	1023	1011	12	0.01
2	1023	1017	6	0.005
3	1023	1015	2	0.001

In table 18 & 19, we can notice the relation between BER and number of operated sensors. The more sensors operate, the less BER goes where result in better signal detection. Also, the number operating sensors vary depends on the underwater channel environment. If there is low level of noise, fewer sensors will be working and then noticeable reduction in the power consumption will be observed**.**

References

[1] I. Akyildiz, D. Pompeili and T. Melodia, (2005), "Underwater Acoustic Sensor Networks: Research Challenges," *Elsevier Journal on Ad Hoc Networks*, Vol. 3, issue 3, pp. 257-279.

[2] Paul Cotae and T.C. Yang "A cyclostationary blind Doppler estimation method for underwater acoustic communications using direct-sequence spread spectrum signals," *8th International conf. on Communications, COMM* 2010, Bucharest, July 2010.

[3] P. Cotae, "Transmitter Adaptation Algorithm for Multicullular Synchronous CDMA Systems with Multipath," *IEEE Journal od Selected Areas in Communications (Special Issue on Next Generation CDMA Technologies)*, vol. 24, no. 1. Pp. 94-103, Jan. 2006.

[4] M. Stojanovic, "On the Relationship between Capacity and Distance in an Underwater Acoustic Communication Channel," *WUWNet*, 2006, pp 41-47.

[5] S. Ali, A. Fakoorian, G. R. Solat, H. Eidi, "Maximizing Capacity in Wireless Sensor Networks by Optimal Placement of Cluster Heads," *Canadian conference on Electrical and Computer Engineering* , May 2008, pp. 001245-001250.

[6] H. Nouri and M. Uysal, "Information Theoretic Analysis and Optimization of Underwater Acoustic Communication Systems," *IEEE*. 978-1-4673-2232. EuroCon 2013.

[7] D. Lucani, M. Stojanovic, M. Medard, "On the Relationship between Transmission Power and Capacity of an Underwater Acoustic Communication Channel", *WUWNet, 2006, pp.41-47.*

[8] M. Stojanovic, "Capacity of a Relay Acoustic Channel," *IEEE Transactions on Communications.* Massachusetts Institute of Technology. 0-933957-35-1. 2007 MTS.

[9] S. Ali, A. Fakoorian, G.R. Solat, H. Taheri, A. Eidi, "Maximizing Capacity in Wireless Sensor Networks by Optimal Placement of Clusterheads," *IEEE,* 978-1-4244-1643, January 2008.

[10] P. Cotae, I. S. Moskowitz and M. H. Kang, "Eigenvalue Characterization of the Capacity of Discrete Memoryless Channels with Invertible Channel Matrices," *Proceedings IEEE 44th Annual Conference on Information Science and Systems*- CISS 2010, Princeton University, March 17-19, pp. 1-6, March 2010.

[11] T. Schurmann and P. Grassberger, "Entropy Estimation of Symbol Sequences," *Department of Theoretical Physics, University of Wuppertal, D-42097 Wuppertal, Germany*.

[12] P. Cotae, S. Yalamanchili, C. L. P. Chen, A. Ayon, "Optimization of Sensor Locations and Sensitivity Analaysis for Engine Health Monitoring Using Minimum Intereference Algorithm," *EURASIP Journal on Advances in Signal Processing.*

[13] T. C. Yang and W. Yang, "Performance Analysis of Direct-Sequence Spread Spectrum Underwater Acoustic Communications with low signal-to-noise ratio input signals*," Journal of Acoustical Society of America*, pp. 842-855, February 2008.

[14] D. Guo, S. Shamai(Shitz) and S. Verdu, "Mutual Information and minimum Mean Square Error in Gaussian Channels," *IEEE Transaction on Information theory*, Vol. 51, no. 4, April 2005.

[15] P. Cotae, "Transmitter Adaptation Algorithm for Multicullular Synchronous CDMA Systems with Multipath," *IEEE Journal od Selected Areas in Communications (Special Issue on Next Generation CDMA Technologies)*, vol. 24, no. 1. Pp. 94-103, Jan. 2006.

[16] R. Cao, F. Qu, L. Yang, "On the Capacity and System Design of Relay-Aided Underwater Acoustic Communications," *IEEE*, 2010, 978-1-4244-6398-5/10.

[17] B. C. Geiger, G. Kubin, "On the Rate of Information Loss in Memoryless Systems*," IEE Signal Processing*, April 2013.

[18] F. Zhao, J. Shin, J. Reich, "Information Driven Dynamic Sensor Collaboration for Tracking Applications," *IEEE Signal Processing Magazine*, Vol. 19, no. 2, pp 61-72, 2002.

[19] A. G. Bessios and F. M. Caimi, "High-rate Wireless data Communciations: An Underwater Acoustic Communications Framework at the Physical Layer," *Department of Electrical Engineering, Harbor Branch Oceanographic Institution, Inc.*, 5600 U.S. 1 North, Fort Pierce, FL 34946.

[20] W. Alsalih, H. Hassanein, and S. Akl, "Placement of multiple mobile data collectors in underwater acoustic sensor networks," *IEEE ICC*, 2008, pp. 2113-2118.

[21] T.M. Cover and J. A. Thomas, '*Elements of Information Theory*,' 2nd ed., New York: Wiley, 2006.

[22] J.G. Proakis, *Digital Communications*, 5th ed., McGraw-Hill, 2008.

[23] E. Ertin, "Maximimum Mutual Information Principle for Dynamic Sensor Query Problems", *IPSN 2003, LNCS 2634*, pp. 405–416, 2003.

[24] R. Shrestha, M. Elsayed, P. Cotae, "On the mutual information of sensor networks in Underwater Wireless Communication: An experimental Approach," *ASEE Zone I Conference, University of Bridgeport, Bridgeport, CT, USA.* April 3-5, 2014.

[25] Mahmoud Elsayed, P. Cotae, "On the Performance of the Underwater Wireless Communication Sensor Networks: Work in Progress" *ASEE Mid-Atlantic Fall 2014 Conference, November 14- 15, 2014, Swarthmore College, Swarthmore, PA, USA.*

[26] Mahmoud Elsayed, P. Cotae, I. Maskowitch "On the Performance of the Underwater Wireless Sensor Networks: An Experimental Approach" *Proceedings of the ASEE Gulf-Southwest Annual Conference, March 25-27, the University of Texas at San Antonio, San Antonio, Taxes, USA.*

[27] MATLAB Signal Processing Toolbox, http://www.mathworks.com/products/signal/

[28] MATLAB Communication toolbox, http://www.mathworks.com/products/communications/

Appendix

MATLAB Simulation Code for the Underwater Acoustic Channel Model

```
%%Noise Calculations
Nt = 1.7-3*log(f);            %%UAC Thermal Noise
Ns = 4+2*(s-0.5)+2.6*log(f)-6*(log(f+0.03));     %%Shipping Noise activity s = 0-->1;
Nth = -1.5 +2*log(f);          %% Noise th.....
Neq= Nt + Ns + Nth;            %%overall PSD of Noise in UAC
%%
f3=0:0.1:1000000;
N= 50 - 18*log10(f3);             %%Noise approximation
plot(f3,loglog(N));
grid

%% Abosrbation Coefficient Calculations

%Thorp Formula (freaquencies above khz)
f2=400:0.01:1000;
loga02= (0.11*(f2.^2)/(1+f2.^2)) + (44*(f2.^2)/(4100+f2.^2))+(2.75*(10^(-4)).*(f2.^2))+0.003;

%foe below few khz
f1=0:0.001:40;
loga01= (0.11.*(f1.^2)./(1+f1.^2))+(0.011.*(f1.^2))+0.002;
loga0= [loga01 loga02];
f=[f1 f2];
figure(1)
plot(f1,loga01);
figure(2)
plot(f2,loga02);
figure(3)
plot(f,loga0);
grid
%% Attenuation Calculation
logA= k*10*log(d)+ d*loga;      %% k=1 -> 2 ; d is distance btw nodes
x=logA;
A=2^x;

%% SNR (gamma) calculation

gamma= Pt/((A)*N);  %Pt power transmitted; A -> inv of 10*logA;

%% Capacity for Underwater Acoustic Channel
% the integral of shannon capacity over the BW
```

```
eq=log2(1+gamma);
C= int(eq, x, f1,f2);        %%Capacity for UWAC (BW= -w --> w_)
c = double (C);
```

MATLAB Code for the Modulation Technique used

```
close all;
clear all;
clc;
SNRdB=1:1:12;                    %Signal to Noise Ratio in dB
SNR=10. ^(SNRdB/10);              %Signal to Noise Ratio in Linear Scale
Bit_Length=10^6;                  %No. of Bits Transmitted
BER_Simulated=zeros(1,length(SNRdB));          %Simulated Bit Error Rate

%Detection Scheme:(Soft Detection)
%+1 if o/p >=0
%-1 if o/p<0
%Error if input and output are of different signs

%% BPSK Transmission over AWGN channel
        parfor  k=1:length(SNR);
                  x=(2*floor(2*rand(1,Bit_Length)))-1;
                  y=(sqrt(SNR(k))*x)+randn(1,Bit_Length);

          BER_Simulated(k)=length(find((y.*x)<0));%Total number of bits in error
        end
        BER_Simulated=BER_Simulated/Bit_Length;
        semilogy(SNRdB,BER_Simulated,'m-<', 'linewidth' ,2.0);
        hold on
        semilogy(SNRdB,qfunc(sqrt(SNR)),'b-','linewidth',2.0);      %Theoritical Bit Error Rate
        title('BPSK over AWGN Simulation');xlabel('SNR in dB');ylabel('BER');
        legend('BER(Simulated)','BER(Theoritical)')
        axis tight
        grid
```

Matlab code for the Information theory aspects tools and Experimental Approach

```
%function [ Ixy, Hx, Hy, Hxy, HyX, HxY] = mySuperFunction(x, y)

%switch nargin
 %   case 2
       % do nothing
  %  case 1
   %     y = x;
```

```
  %  otherwise
   %     clc
    %    clear all
     %   load ('data.mat');
%end

%**********Events Probablility Distribution***********
%Event A--> P(x=1,y=1) ; Event B--> P(x=0, y=0);
%Event C--> P(x=1,y=0) ; Event D--> P(x=0, y=1);
%Event G--> P(y~=0,y~=1) No Joint Prabality;(Bit Loss)

n=1023;     %Number of transmitted bits
w=x+y-1;
z=x-y;

A = sum(w(w==1));       % x=1 -> y=1
B = -1 * sum(w(w==-1)); % x=0 -> y=0
C = sum(z(z==1));       % x=1 -> y=0
D = -1 * sum(z(z==-1)); % x=0 -> y=1
G = n - A - B - C - D;

%%
%Joint Probabilities
PA= A/n;      %P(y=1,x=1);
PB= B/n;      %P(y=0,x=0);
PC= C/n;      %P(y=0,x=1);
PD= D/n;      %P(y=1,x=0);
PG= G/n;      %P(y~=1,y~=0);

%Since binary data;
% Estimating the P(x=1) and P(x=0);
Px1 = sum(x(x==1))/n;
Px0 = 1 - Px1;

% Estimating the P(y=1), P(y=0) and P(y~=0 && y~=1);
v = y + 1;
Py1 = sum(y(y==1))/n; % P(y=1)-->Probability of receiving 1;
Py0 = sum(v(v==1))/n; % P(y=0)-->Probability of receiving 0;
Py_ = 1 - Py0 - Py1;  % P(y!=0,1)-->Probability of receiving null;


% Conditional Probablilities
Py1X1= PA/Px1;   %P(y=1|x=1)= P(y=1,x=1)/P(x=1);
Py0X0= PB/Px0;   %P(y=0|x=0)= P(y=0,x=0)/P(x=0);
Py0X1= PC/Px1;   %P(y=0|x=1)= P(y=0,x=1)/P(x=1);
Py1X0= PD/Px0;   %P(y=1|x=0)= P(y=1,x=0)/P(x=0);
```

```
Py_X = PG/(Px0 + Px1);   %P(y=?|x=0,1)= P(y=1,x=0)/ (P(x=0) + P(x=1));


 %%
 % Entropy, Joint Entropy and Mutual information Calculation

 % H(x) Entropy(transmitted bits)

  Hx = 0;
  if Px1 ~= 0
     Hx = Hx + Px1*log2(1/Px1);
  end

  if Px0 ~= 0
     Hx = Hx + Px0*log2(1/Px0);
  end

 % H(y) Entropy(Received bits);

  Hy = 0;
  if Py1 ~= 0
     Hy = Hy + Py1*log2(1/Py1);
  end

  if Py0 ~= 0
     Hy = Hy + Py0*log2(1/Py0);
  end

  if Py_ ~= 0
     Hy = Hy + Py_*log2(1/Py_);
  end

  % H(x,y) Joint Entropy

  Hxy = 0;

  if PA ~= 0
     Hxy = Hxy + PA*log2(PA);
  end

  if PB ~= 0
     Hxy = Hxy + PB*log2(PB);
  end

  if PC ~= 0
     Hxy = Hxy + PC*log2(PC);
```

```
    end

    if PD ~= 0
        Hxy = Hxy + PD*log2(PD);
    end

    if PG ~= 0
        Hxy = Hxy + PG*log2(PG);
    end

    Hxy = -Hxy;

    % H(x|y) Conditional Entropy
    HxY = Hxy - Hy;     %Conditional Entropy H(x|y) = H(x,y) - H(y);
    HyX = Hxy - Hx;     %Conditional Entropy H(y|x) = H(x,y) - H(x);

    Ixy = Hx - HxY;     % I(x,y) Mutual Information between Transmiter and Receiver

%end

%% Maximum Ratio Combining and BER Threshold
Yc=char();
for i=1:10
    m= [sum(y2(i,:)), sum(y3(i,:)), sum(y4(i,:))];
    if max(m)==sum(y2(i,:))
        Yc(i,:)=y2(i,:);
    elseif max(m)==sum(y3(i,:))
        Yc(i,:) = y3(i,:);
    else Yc(i,:) = y4(i,:);
    end
end
Yct=cellstr(Yc);
y=Yct';
%%
% PowerCalculations
if BER < TH1
    %s2 --> on
end %end and loop back to find new BER
```

MATLAB Code for the GUI (SAM Control)

```
function varargout = sam(varargin)
% GUI MATLAB code for gui.fig
```

```
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 17-Jun-2014 16:53:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
handles.output = hObject;
handles.comport = 5;
handles.inputbuffer = 32;
```

```
handles.outputbuffer = 32;
handles.thresh = 50;
handles.TXspeed = 'S4';
handles.RXspeed = 'R4';
handles.timeout = 10;
handles.message = 'Test Message 0123456789';
handles.howMany = 4;
handles.pulselength = 2;
handles.filename = 'filename';
handles.howMany_RX = 4;
handles.RXTime = 20;
handles.TXTime = 20;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
clc
clear all

end

% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
end

% CONNECT TO COM PORT BUTTON
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

comport_Full = strcat('COM', num2str(handles.comport));

% Find a serial port object.
s = instrfind('Type', 'serial', 'Port', num2str(comport_Full), 'Tag', '');
% Create the serial port object if it does not exist..
% ..otherwise use the object that was found.

if isempty(s)
    s = serial(num2str(comport_Full));
     else
    fclose(s);
    s = s(1);
end
```

```
ipt = handles.inputbuffer;
opt = handles.outputbuffer;
tim = handles.timeout;


%configure instrument object,s
set(s,'BaudRate',4800);
set(s,'DataBits',8);
set(s, 'FlowControl', 'software');
set(s,'Timeout',tim);
set(s,'InputBufferSize',ipt);
set(s,'OutputBufferSize',opt);

%display('baud rate is '); num2str(get(s,'BaudRate'))
%display('data bits '); num2str(get(s,'DataBits'))
%display('flow control is '); num2str(get(s, 'FlowControl'))
%display('timeout is '); num2str(get(s,'Timeout'))
%display('input buffer is '); num2str(get(s,'InputBufferSize'))
%display('output buffer is '); num2str(get(s,'OutputBufferSize'))

handles.s = s;
handles.comportfull = comport_Full;

guidata(hObject, handles);

fopen(s);
s
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


function comport = edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double
```

```
comport = get(hObject,'String');

handles.comport = comport;

% Save the handles structure.
guidata(hObject,handles)
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double

inputbuffer = get(hObject,'String');

handles.inputbuffer = inputbuffer;

% Save the handles structure.
guidata(hObject,handles)

end

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
```

```
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2
end

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
end

% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4
end

% --- Executes on button press in radiobutton5.
```

```
function radiobutton5_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton5
end

% --- Executes on button press in radiobutton6.
function radiobutton6_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton6

end
% --- Executes on button press in radiobutton7.
function radiobutton7_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton7
end

% --- Executes on button press in radiobutton8.
function radiobutton8_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton8
end

% --- Executes on button press in radiobutton9.
function radiobutton9_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton9
end

% --- Executes on button press in radiobutton10.
function radiobutton10_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton10
end

% --- Executes on button press in radiobutton11.
```

```
function radiobutton11_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton11
end

% --- Executes on button press in radiobutton12.
function radiobutton12_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton12
end

% --- Executes on button press in radiobutton13.
function radiobutton13_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton13
end

% --- Executes on button press in radiobutton14.
function radiobutton14_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton14
end

% --- Executes on button press in radiobutton15.
function radiobutton15_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton15
end


function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double
```

```
thresh = get(hObject,'String');
if str2num(thresh)<1
   errordlg('Value must be from 1 - 99.','Threshold Error');
   set(handles.edit6,'String', '1');
elseif str2num(thresh)>99
   errordlg('Value must be from 1 - 99.','Threshold Error');
    set(handles.edit6,'String', '99');
else


handles.thresh = thresh;

% Save the handles structure.
guidata(hObject,handles)
end
end

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double
end

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s = handles.s;

oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;

% your computation goes here

y = get(s,'BytesAvailable');
if y == 0
    set(handles.text16,'String','<Buffer is empty>');
else
y = fscanf(s);
set(handles.text16,'String',y);
end
set(handles.figure1, 'pointer', oldpointer)


end


function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double

message = get(hObject,'String');

handles.message = message;

% Save the handles structure.
guidata(hObject,handles)

end

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

message = handles.message;
s = handles.s;

oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;


fwrite(s,message,'int8');

set(handles.figure1, 'pointer', oldpointer)


end

% --- Executes during object creation, after setting all properties.
function pushbutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


thresh_Full = strcat('T', num2str(handles.thresh));

s = handles.s;

fprintf(s,num2str(thresh_Full))
```

```
response = fscanf(s)
set(handles.text25,'String',response);
end

% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

s = handles.s;

fprintf(s,num2str(handles.TXspeed))
response = fscanf(s)
set(handles.text25,'String',response);
end

% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

s = handles.s;

fprintf(s,num2str(handles.RXspeed))
response = fscanf(s)
set(handles.text25,'String',response);
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
togglebutton1.
function togglebutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end

% --- Executes on key press with focus on togglebutton2 and none of its
controls.
function togglebutton2_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)


end
```

```
% --- Executes on key press with focus on togglebutton1 and none of its
controls.
function togglebutton1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  structure with the following fields (see UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)


end


% --- Executes when selected object is changed in uipanel3.
function uipanel3_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel3
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
s = handles.s;
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.

case 'togglebutton1'
fprintf(s,'%s','###')
response = fscanf(s,'%s',4)
set(handles.text25,'String',response);

case 'togglebutton2'
fprintf(s,'%s','D')
response = fscanf(s,'%s',5)
set(handles.text25,'String',response);

end

end


% --- Executes when selected object is changed in uipanel1.
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel1
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.

case 'radiobutton2'
    handles.TXspeed = 'S0';
case 'radiobutton3'
```

```
    handles.TXspeed = 'S1';
case 'radiobutton7'
    handles.TXspeed = 'S2';
case 'radiobutton4'
    handles.TXspeed = 'S3';
case 'radiobutton6'
    handles.TXspeed = 'S4';
case 'radiobutton5'
    handles.TXspeed = 'S5';
case 'radiobutton8'
    handles.TXspeed = 'S6';
end

guidata(hObject, handles);

end


% --- Executes when selected object is changed in uipanel2.
function uipanel2_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel2
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.

case 'radiobutton9'
    handles.RXspeed = 'R0';
case 'radiobutton10'
    handles.RXspeed = 'R1';
case 'radiobutton14'
    handles.RXspeed = 'R2';
case 'radiobutton11'
    handles.RXspeed = 'R3';
case 'radiobutton13'
    handles.RXspeed = 'R4';
case 'radiobutton12'
    handles.RXspeed = 'R5';
case 'radiobutton15'
    handles.RXspeed = 'R6';
end

guidata(hObject, handles);
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double
end

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end

% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s = handles.s;
fclose(s);
s
delete(s)
clear s
end


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double

inputbuffer = str2double(get(hObject,'String'));

handles.inputbuffer = inputbuffer;

% Save the handles structure.
guidata(hObject,handles)
end


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end


function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a
double

outputbuffer = str2double(get(hObject,'String'));

handles.outputbuffer = outputbuffer;

% Save the handles structure.
guidata(hObject,handles)
end


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton20.
function pushbutton20_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a
double
timeout = str2double(get(hObject,'String'));

handles.timeout = timeout;

% Save the handles structure.
guidata(hObject,handles)

end

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
end


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes on button press in pushbutton24.
function pushbutton24_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s = handles.s
howMany = handles.howMany
pulselength = handles.pulselength

n = zeros(1,pulselength); % preallocate variable n
for g = 1:pulselength;
  n(g) =  (-1).^g;    % generate pulse
end

for h = 1:str2num(howMany);
   fwrite(s, n,'int8'); % send pulse number h
   myWait(1);

end
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a
double
pulselength = str2double(get(hObject,'String'));

if pulselength<1
   errordlg('Value must be from 1 - 1024.','Pulse Length Error');
   set(handles.edit19,'String', '1');
elseif pulselength>1024
   errordlg('Value must be from 1 - 1024.','Pulse Length Error');
    set(handles.edit19,'String', '1024');
else
```

```
handles.pulselength = pulselength;

% Save the handles structure.
guidata(hObject,handles)
end
end

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a
double
howMany = str2double(get(hObject,'String'));

if howMany<1
   errordlg('Value must be from 1 - 100.','Error');
   set(handles.edit20,'String', '1');
elseif howMany>100
   errordlg('Value must be from 1 - 100.','Error');
    set(handles.edit20,'String', '100');
else


handles.howMany = howMany;

% Save the handles structure.
guidata(hObject,handles)
end
end

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end


% --- Executes on button press in pushbutton25.
function pushbutton25_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;
s = handles.s;
filename = handles.filename;
filename_Full = strcat(num2str(filename),'.xlsx');
howMany_RX = handles.howMany_RX;
RXTime = handles.RXTime;



x = get(s,'BytesAvailable');
if x == 0
    errordlg('Buffer is empty.','Buffer Error');
else
    set(handles.text32,'String',RXTime);
    for g = 1:howMany_RX
        set(handles.text31,'String',howMany_RX-g);
        y{g} = fscanf(s)
        y{g}
        if g ~= howMany_RX
            for h = 1:RXTime
                pause(1)
                j = RXTime-h;
                set(handles.text32,'String',j);
            end
        else
        end
    end
end
pause(.2)

y = strtrim(y);

assignin('base','y',y)
xlswrite(filename_Full,y)
winopen(filename_Full)

set(handles.figure1, 'pointer', oldpointer)
end
```

```
function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a
double
filename = get(hObject,'String');

handles.filename = filename;

% Save the handles structure.
guidata(hObject,handles)
end

% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a
double

message = get(hObject,'String');

handles.message = message;

% Save the handles structure.
guidata(hObject,handles)

end

% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in pushbutton26.
function pushbutton26_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

message = handles.message;
s = handles.s;

oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;

if mod(length(message),2) == 0
    %number is even
    message = [message,' ']
    else
    message = message
end

fprintf(s,'%s\n',message);


set(handles.figure1, 'pointer', oldpointer)


end

% --- Executes on button press in pushbutton27.
function pushbutton27_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
oldpointer = get(handles.figure1, 'pointer');
set(handles.figure1, 'pointer', 'watch')
drawnow;
s = handles.s;
TXTime = handles.TXTime;
howMany = handles.howMany;
pulselength = handles.pulselength;

n = zeros(1,pulselength); % preallocate variable n
for g = 1:2:pulselength;
  n(g) =  1;    % generate pulse
```

```
end

n = sprintf('%d',n);

set(handles.text30,'String',TXTime);

for g = 1:howMany

set(handles.text29,'String',howMany-g);

if mod(length(n),2) == 0
    %number is even
    n = [n,' '];
    else

end
fprintf(s,'%s\n',n);
pulse = n
if g ~= howMany
for h = 1:TXTime
    pause(1)
    j = TXTime-h;
    set(handles.text30,'String',j);
end

else
end
end


set(handles.figure1, 'pointer', oldpointer)
end


function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23 as a
double
pulselength = str2double(get(hObject,'String'));

if pulselength<1
   errordlg('Value must be from 1 - 1024.','Pulse Length Error');
   set(handles.edit19,'String', '1');
elseif pulselength>1024
   errordlg('Value must be from 1 - 1024.','Pulse Length Error');
    set(handles.edit19,'String', '1024');
else


handles.pulselength = pulselength;
```

```
% Save the handles structure.
guidata(hObject,handles)
end
end
% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24 as a
double
howMany = str2double(get(hObject,'String'));

if howMany<1
   errordlg('Value must be from 1 - 100.','Error');
   set(handles.edit20,'String', '1');
elseif howMany>100
   errordlg('Value must be from 1 - 100.','Error');
    set(handles.edit20,'String', '100');
else


handles.howMany = howMany;

% Save the handles structure.
guidata(hObject,handles)
end

end

% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end


% --- Executes during object creation, after setting all properties.
function uipanel3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes during object creation, after setting all properties.
function uipanel7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes during object creation, after setting all properties.
function text16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end


% --- Executes during object creation, after setting all properties.
function text25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
end



function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25 as a
double
TXTime = str2double(get(hObject,'String'));

if TXTime<1
   errordlg('Value must be from 1 - 9999.','Time Error');
   set(handles.edit25,'String', '1');
```

```
elseif TXTime>9999
   errordlg('Value must be from 1 - 9999.','Time Error');
    set(handles.edit25,'String', '9999');
else


handles.TXTime = TXTime;

% Save the handles structure.
guidata(hObject,handles)
end
end


% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26 as a
double
howMany_RX = str2double(get(hObject,'String'));

if howMany_RX<1
   errordlg('Value must be from 1 - 100.','Error');
   set(handles.edit26,'String', '1');
elseif howMany_RX>100
   errordlg('Value must be from 1 - 100.','Error');
    set(handles.edit26,'String', '100');
else


handles.howMany_RX = howMany_RX;

% Save the handles structure.
guidata(hObject,handles)
end
```

```
end

% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end



function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27 as a
double
RXTime = str2double(get(hObject,'String'));

if RXTime<1
   errordlg('Value must be from 1 - 9999.','Time Error');
   set(handles.edit27,'String', '1');
elseif RXTime>9999
   errordlg('Value must be from 1 - 9999.','Time Error');
    set(handles.edit27,'String', '9999');
else


handles.RXTime = RXTime;

% Save the handles structure.
guidata(hObject,handles)
end
end

% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
end
```